(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 329 023 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: **89102276.6**

(51) Int. Cl.[4]: **G06F 15/332**

(22) Date of filing: **10.02.89**

(30) Priority: **16.02.88 US 155671**

(43) Date of publication of application:
23.08.89 Bulletin 89/34

(84) Designated Contracting States:
DE FR GB NL

(71) Applicant: **HONEYWELL INC.**
Honeywell Plaza
Minneapolis Minnesota 55408(US)

(72) Inventor: **Magar, Surendar S.**
2925 Orion Drive
Colorado Springs, CO 80906(US)
Inventor: **Fleming, Michael E.**
10385 Raygor Road
Colorado Springs, CO 80908(US)
Inventor: **Shen, Shannon N.**
60 Broadmoor Hills Drive
Colorado Springs, CO 80906(US)
Inventor: **Rishavy, Kevin M.**
6230 Pemberton Way
Colorado Springs, CO 80919(US)
Inventor: **Furman, Christopher D.**
2715 Reeve Circle, No. 1018
Colorado Springs, CO 80906(US)
Inventor: **Murphy, Kenneth N.**
5420 Mule Deer Drive
Colorado Springs, CO 80919(US)

(74) Representative: **Rentzsch, Heinz et al**
Honeywell Europe S.A. Holding KG Patent & License Dept. Postfach 10 08 65
Kaiserleistrasse 39
D-6050 Offenbach am Main(DE)

(54) Apparatus and method for performing digital signal processing including fast fourier transform radix-4 butterfly computations.

(57) A digital array signal processor and associated method are described for implememting the fast Fourier transform radix-4 butterfly algorithm. The digital array signal processor is an integrated circuit with a four stage pipeline and can perform a radix-4 butterfly operation on four complex operands every 80 nanoseconds. Using the decimation-in-frequency implementation of the radix-4 butterfly algorithm, the digital array signal processor includes a first stage for distribution of complex input operand values, a second stage for performing addition and subtraction operations, a third stage for performing multiplication operations and a fourth stage for distribution of the output operand values. The digital array signal processor can be reconfigured to perform a radix-2 butterfly operation on two sets of two complex numbers during the 80 nanosecond machine cycle. The digital signal array processor can be configured to perform a series of operations on an array of operands or can be one of a series of processors, each processor performing a separate operation on an operand array.

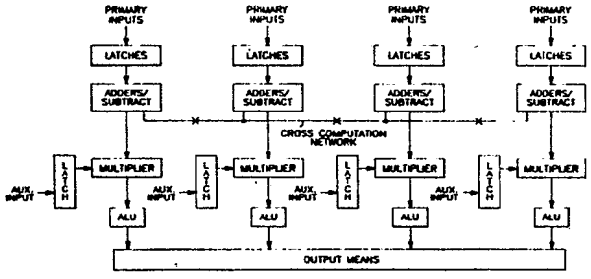


Fig. 3.1

EP 0 329 023 A2

APPARATUS AND METHOD FOR PERFORMING DIGITAL SIGNAL PROCESSING INCLUDING FAST FOURIER TRANSFORM RADIX-4 BUTTERFLY COMPUTATIONS

1.0 INTRODUCTION

The Discrete Fourier Transform (DFT) is commonly used in a variety of Digital Signal Processing (DSP) Systems. The Fast Fourier Transform (FFT) algorithm is frequently used for computing DFT due to its computation efficiency. Two VLSI devices are described for applications in high performance FFT and other DSP systems. Both devices will utilize 1.2 u CMOS dual-level-metal process.

One of the devices, named Computational Element (CE) provides a high speed computing engine for DSP systems. The other device, termed Micro Controller (MC), integrates various address generators and controlling functions, which are commonly used in the FFT systems. Extremely fast and self contained systems can be designed by combining CE and MC devices with off-the-shelf memory devices. The system performance can be traded against choosing the number of CEs, MCs and memory devices in a given system. For example, the chip-set can implement FFT systems ranging from 4 complex-points to 64K complex-points. A 1024 complex-point FFT, for example, can be performed in 20 microseconds (us) using the disclosed hardware.

This description emphasizes the characteristics of the CE device. However, a brief introduction of both the CE and MC, and their applications to the FFT systems, as an example, will be presented briefly in this chapter.

1.1 CE - INTRODUCTION

The CE is a highly integrated, application specific, semi-programmable, computational device containing several multipliers, adders and registers. The device can perform a variety of FFT-specific and general purpose DSP operations, including logical operations at extremely high speed. For example, a radix-4 butterfly or, a block of four complex-multiples, can be performed in 80 ns (the machine cycle of the computational element) in a pipelined mode. The machine cycle of the device, which is specified to be 80 ns, is termed Tm. Basically, the device accepts two sets of four ccmplex-data values and produces one set of four complex-data values every machine cycle, in a pipelined mode. Its block floating point 16 bit parallel architecture is optimized to implement following FFT specific functions on a single chip:

a. Full radix-4 butterfly: A single column of the radix-4, N complex-point FFT takes N/4 * Tm ns.

b. Two full radix-2 butterflies: As radix-4, a single column of radix-2, N complex-point FFT takes N/4 * Tm ns.

c. Block Multiply on four pairs of complex-data values: It is useful for windowing, complex modulation/demodulation, etc. It takes a total of N/4 * Tm ns to window or multiply N complex-data values. The windowing operations are typically performed prior to FFT.

d. Square-Sum four complex-data values: As before, it takes a total of N/4 * Tm ns to square-sum N complex data values. Such operations are normally required after the FFT for magnitude-square computation.

e. Trigonometrical recombination to implement two N real-point FFTs on a N complex-point FFT machine: Once again, it takes N/4 * Tm ns to produce two N real-point FFTs from a N complex-point FFT.

f. Trigonometrical recombination to implement a 2N real-point FFT from a N complex-point FFT: In this case, the CE operates on two complex data values and produces one complex data value. Therefore, it takes a total of N * Tm ns to produce a 2N real-point FFT from a N complex-point FFT.

In addition, the CE has a variety of general purpose functions available such as Block-Add, Block-Subtract, Block-AND, Block-EXOR. These functions operate on blocks of four complex values every machine-cycle.

1.1.1. CE - Input/Output

As stated in the above section, four complex data values flow through the CE every machine cycle. All the data input and output operations on the CE are performed over multiple 16 bit parallel buses. There are two different bond-out versions of the CE to support two types of external bus/memory system architectures

as shown in Figure 1.1. The dual I/O CE is configured around four data buses called D1, D2, D3 and D4. It also has two auxiliary data input buses AXD1 and AXD2 for feeding twiddle (scaling and constant) factors, etc. At any time, the pair D1 and D2 could be receiving the data while the pair D3 and D4 could be outputting data. The role of this pair of buses could be reversed at any machine cycle via an instruction. The dual I/O CE accepts a complex value every Tm/4 ns, over each pair of its input buses. Therefore, a set of four complex values are inputted every Tm ns over input bus pairs. Similarly, the device outputs a complex value on its pair of output buses every Tm/4 ns, producing a set of four complex values every Tm ns. At maximum clock rate (Tm = 80 ns), each bus operates a 50 MHz (20 ns per 16 bit transfer). A system operating at such rate will require memories with access time around 10 ns which is prohibitive at present. Practically, a system can be designed with buses operating at 33.3 MHz (30 ns per transfer, Tm = 120 ns) using state of the art memories. A 50 MHz system will become possible in the next couple of years.

The quad I/O version of the CE will be available in a larger package. The quad I/O CE, which is shown in Figure 1.1, contains four data input buses (D1, .D2, D5, D6) and four data output buses (D3, D4, D7, D8). The twiddle factors, etc. are fed over four auxiliary buses AXD1, AXD2, AXD3, AXD4. In this case, each bus is operated at a rate of 25 MHz since the number of buses have been doubled. In the first half of the machine cycle, four real inputs are fed to the device. The corresponding four imaginary inputs are fed in the latter half of the machine cycle. The outputs are produced in the same sequence on the output buses. The role of the input and output buses can not be reversed in the quad I/O mode.

The dual I/O version of the CE could be used to design very flexible FFT and other DSP systems by using off-the-shelf single-port memories. This flexibility is possible because the dual I/O CE transfers one complex value at a time, the real and imaginary part of which are stored at the same address. In addition, its lower pin count (144 pin package) makes it very attractive for system design. Initially, the device can be packaged in a PGA package and a surface mount version will be made available at a later stage. On the other hand, quad version of the CE is attractive due to somewhat relaxed memory access time. However, it leads to a larger package (244 pins) which makes it prohibitive in a surface mount package at this time. Also, a four port memory system may be required when standard FFT algorithms are used because it deals with parts of four complex numbers at a time.

1.1.2 CE - Other Architectural Aspects

Apart from different I/O configurations, both versions of the CE have identical internal data paths. A series of magnitude-detectors are provided at the output data stages of the CE which continuously monitor the upper 6 bits of the output data array for FFT operations. A scaling factor is produced after processing a column of N complex-point data array. The scaling factor is used to scale the data array on the input stage of the next pass by the input scalers (shift and round mechanism), included on the chip. It protects the CE adders from overflowing during the computation by providing conditional scaling. An accumulated scale factor is available at the beginning of each pass, which should be effectively treated as an exponent for the normalization of the processed array of data. The scheme is referred to as Block Floating Point in the literature.

1.2 MC - INTRODUCTION

The FFT Micro Controller (MC) is a device which provides all the addressing sequences required to address the memory devices associated with an FFT system. The MC is intended to be a companion device for the dual I/O version of the CE device for FFT applications only. It generates the necessary addressing sequences and control signals for the dual I/O CE and associated memories minimizing external glue logic. Typical FFT systems can be built by using the CE, the MC and off-the-shelf memories. The MC is designed to generate the following addressing sequences supporting FFT systems up to 64K complex-points.

a. Data and twiddle factor addresses for radix-4, in-place Decimation-in-Frequency (DIF) FFT algorithm.

b. Data memory and twiddle factor addresses for radix-2, in-place DIF FFT algorithm.

c. Sequential addresses which are useful for windowing and square-summing, etc.

d. Addresses required for sequencing data for trigonometrical recombination associated with formation of 2N real-point FFT from a N complex-point FFT.

e. Addresses required for sequencing data for trigonometrical recombination associated with formation of two separate N real-point FFTs from a N complex-point FFT.

f. Sequential addresses to address input-data collection memory and output data memory. The output address may be digit-reversed if desired.

SUMMARY OF THE INVENTION

The foregoing and other features are accomplished, according to the present invention, by providing a computational element, responsive to control signals, for performing a plurality of digital signal processing operations on the elements of an operand array. In response to control signals, the computational element can perform a fast Fourier transform radix-4 butterfly computation on four complex operands every 80 ns in a pipelined mode. In response to control signals, the computational element can be configured to perform two sets of fast Fourier transform radix-2 butterfly computations. The computational element can be further configured to perform arithmetic and logic operations on an operand array. The computational element can be configured to perform a plurality of sequential operations on an operand array or can be configured to perform one of a sequence of operations when coupled in series with at least one other computational element. The computational element of the present invention can be fabricated on a single chip using integrated circuit techniques. A microcontroller, providing control signals and constant factors for computation to the computational element, can be fabricated on a second chip.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1.1 displays the signals related to the dual I/O computational element and the quad I/O computational element according to the present invention.

Figure 1.2 displays the signals related to the microcontroller according to the present invention.

Figure 1.3 is a block diagram of a dual memory fast Fourier transform system according to the present invention.

Figure 1.4 is a block diagram of a cascaded memory fast Fourier transform system.

Figure 1.5 is a block diagram of a one memory fast Fourier transform system according to the present invention.

Figure 1.6 is a block diagram of reduced configuration non-real time fast Fourier transform system according to the present invention.

Figure 2.0 is a block diagram of the components comprising the computational element according to the present invention.

Figure 2.1 displays the input and output operations performed over pairs of buses in the dual I/O configuration.

Figure 2.2 displays the input and output signals for the dual I/O configuration in two direction modes.

Figure 2.3 displays the input and output signals for the computational element using two sets of operand terminals as both input and output interface paths.

Figure 2.4 displays the signals of the computational elememt in the quad I/O mode.

Figure 2.5 is a block diagram of the internal clock generator of the computational element.

Figure 2.6 is a timing diagram for internal clock signals related to the dual I/O configuration of Fig. 2.1 and Fig. 2.2.

Figure 2.7 is a timing diagram for internal clock signals generated for the dual I/O configuration of Fig. 2.3.

Figure 2.8 is a timing diagram for internal clock signals generated in the quad I/O configuration of Fig. 2.4.

Figure 2.9 is a timing diagram for I/O signals in the dual I/O configurations of Fig. 2.1 and Fig. 2.2.

Figure 2.10 is a timing diagram for I/O signals in the dual I/O configuration of Fig. 2.3.

Figure 2.11 is a timing diagram for I/O signals in the quad I/O configuration of Fig. 2.4.

Figure 2.12 is a block diagram of the components of the I/O stages of the computational element, with Figures 2.12a to 2.12d located side-by-side in one row and Figures 2.12e to 2.12g positioned above Figures 2.12b to 2.12d.

Figure 2.13 is a block diagram of components of processor A of the computational element with Figures 2.13a and 2.13b positioned side-by-side.

Figure 2.14 is a block diagram of the components of processor B of the computational element with parts 2.14a and 2.14b arranged in an upper row side-by-side and parts 2.14c and 2.14d arranged in a lower row beneath parts 2.14a and 2.14b, respectively.

Figure 2.15 is a timing diagram for processor B.

Figure 3.1 is a block diagram of component groups for the computational element.

Figure 3.2 is a block diagram of component groups for a second embodiment of the computational element.

Figure 3.3 illustrates the decimation-in-time algorithm.

1.2.1 MC-Architecture

A block diagram of the MC is shown in Figure 1.2 illustrating various buses and control signals. The device produces five 16 bit address outputs, labeled as IA(15:0), D1A(15:0), AXA(15:0), D2A(15:0) and OA-(15:0). Each bus produces addresses at a rate of 50 MHz, matching the speed of dual I/O CE device. The IA bus produces sequential addresses needed to collect the input data while previous data is being transformed. The D1A and D2A buses produce read-data and write-data addresses corresponding to the FFT algorithm. Since an in-place DIF FFT algorithm is employed by the MC, the read-address and the write-address patterns are the same but separated by a latency-factor associated with the CE and the rest of the system. The twiddle factor memory (auxiliary data memory) address sequence is produced on the AXA bus. A sequential or digit-reversed address can be produced on the OA bus to read the output data memory, containing previously transformed data. In addition, the device produced various signals to control external bus-drivers, memory and the CE.

The device also has a host interface to communicate to a host microprocessor or a control memory. The host processor could use the IA bus as an address bus to address internal registers during initialization. The corresponding data is transferred over D1A bus in this case. If a host processor is not employed in the system, the MC is capable of down loading itself from an external memory for stand-alone applications. Following are some of the registers of the MC which are down loaded initially:

a. FFT size register

b. Latency factor register- determines the latency (due to the CE and external intermediate latches) between Read and Write addresses.

c. Instruction Register File, etc.

The instruction register file contains a set of instructions to implement the desired FFT system configuration. A normal FFT system is composed of multiple "PASSES" of the N complex-point numbers through the CE; the PASSES being windowing, FFT nodes, Square-Sum, etc. One instruction per PASS is executed by the MC to appropriately manage the external memories and the CE. Each instruction contains information such as the following:

a. A code to route the outputs of internal address generators to appropriate external MC address pins.

b. A code which is directly outputted on MC pins to control the CE, external memories, switches, etc.

The MC is packaged in a 132 pin package, dissipating less than 500 mW. The device is available in a PGA package. A surface mount version will be made available at a later stage.

1.3 FFT SYSTEM CONFIGURATIONS

The MC and the CE can be used in a variety of system configurations providing various trade-offs between performance, hardware and cost. Following are some of the system configurations which could be produced by using the CE and the MC. The CE can be used in a variety of non-FFT systems by using the system architectures discussed here.

1.3.1 Mid-Performance System

A N complex-point FFT system is shown in Figure 1.3 which is based upon a dual ping-pong memory architecture. Independent input and output buffer memories have been included for real time operation. At a given instance, while new data is being collected in memory A, previously transformed data is being read from output memory D. The IA and OA buses of MC supply the input and output memory addresses respectively. Meanwhile, current data is being transformed by the CE, supported by dual memories B and C. The data flows in a ping-pong fashion between memories B anc C for multiple passes. Each pass of the N complex-point data can perform full windowing, or one column of the FFT, or square-summing, etc. At each pass, address buses of the MC can be programmed to switch in a ping-pong fashion. In addition, the CE is also capable of interchanging its input/output data ports. Therefore, minimum external bus multiplexers are required. After a complete transform, the external data buses are switched around to interchange the role of the memories. It takes log4(N) passes through the CE to perform a N complex-point FFT using a radix-4 algorithm. Additional passes are needed for windowing and square/summing if so desired. Note that each pass takes memory cycles (N/4 machine cycles). A memory cycle is defined to be a read or a write operation.

1.3.2 High-End System

By employing one CE, one MC and two buffer memories for every node, a cascaded FFT system could be produced as shown in Figure 1.4. The figure shows a N complex-point FFT system, illustrating M nodes. The first node is an input node performing windowing operation. The last node is an output node dedicated to square-summing. Intermediate nodes are dedicated to the FFT algorithm, each node corresponding to a column of FFT. Each FFT system node is double buffered with N complex-point data memories so that data flows through the CE as described in the previous section but in one direction. After completing each pass, the paired buffer memories are interchanged, maintaining a data flow from left to right continuously. Each MC can be programmed to switch internally the addresses on the buses D1A and D2A, etc. to match the interchanging of the memories at the end of each pass. The performance of such a FFT system corresponds to the time taken for one N complex-point pass (N memory cycles).

1.3.3 Low-End System

Since the MC employs an in-place FFT algorithm, FFT systems could be produced by using one memory system. A block diagram of such a system is shown in Figure 1.5. Two additional memory systems have been included for the data-input and data-output maintaining a real-time operation. At any time, only one memory is connected to the CE. Both the read and the write operations have to be performed to that memory for every complex-point of data (two memory cycles). The MC automatically manages address buses for alternate read and write operations. It takes approximately 2N memory cycles for every pass of N complex-point data. Therefore, a total of 2N of log 4(N) memory cycles will be required for a N complex-point FFT.

1.3.4 Low-End Non-Real Time System

The lower-end system, described previously, can be collapsed to minimum hardware configuration by discarding the input and output memories as shown in Fig 1.6. In this case, the MC can be programmed first to collect the input data in the data RAM. Then, data can be processed by making multiple passes through the CE. Finally, the data can be output to an external device. The MC can be programmed to generate all the address sequences on the D1A bus. The performance of such a system is similar to the previous system. However, additional time will be taken for the input and output operations.

1.3.5 FFT System Performance

A summary of FFT system performance, based on the CEs and the MCs is given in Table 1.1. The mid-range system offers a middle compromise between the amount of hardware and performance. It requires one CE, one MC, 24 memory chips and few bus drivers to implement a real-time FFT system, including the input/output data memories. The application of CE and MC is not limited to the system

configurations here. Many other higher performance and intermediate performance configurations are possible due to the programmable nature of the MC and CE.

1.4 QUAD-MODE MC

The MC has been primarily designed to support the dual I/O CE for in-place, DIF, radix-4/radix-2 FFT systems. In dual I/O system, the MC is capable of producing four addresses per machine-cycle/per address-bus. However, four strobe signals have been provided to latch those four addresses into external registers in a machine cycle. The external registers could be used to address a four-port memory system, supporting the quad I/O CE.

TABLE 1.1

FFT - PERFORMANCE (APPROXIMATE)			
	HIGH-END SYSTEM	MID-RANGE SYSTEM	LOWER-END SYSTEM
N-COMPLEX POINT FFT	N*M	N*M*LOG$_4$N + N*M*K	2N*M*LOG$_4$N + 2N*M*K
N = 1024 COMPLEX-POINTS, M = 20 ns K = 0 (NO WINDOW, SQ/SUM)	20•48 µs	102•4 µs	204•8 µs
N = 1024 COMPLEX-POINTS, M = 20 ns K = 2 (WINDOW, SQ/SUM)	20•48 µs	143•36 µs	286•72 µs
NOTE:			
M = MEMORY CYCLE-TIME (READ OR WRITE) N = NUMBER OF COMPLEX DATA POINTS K = ADDITIONAL PASSES FOR WINDOWING, SQUARE/SUM, ETC.			

2.0 CE- ARCHITECTURE

The Computational Element (CE) can be viewed as a very high performance, semi-programmable, application-specific, vector processor which operates on arrays of data at very high speed. Although it has been primarily designed for FFT and related systems, its semi-programmable architecture lends itself in a variety of other DSP applications such as FIR filters, DFTs, complex-modulation/demodulation, etc. Its programmability, performance and cost-effectiveness allows it to be used for implementing time-domain DSP systems by using frequency-domain DSP techniques. The CE also has the capability of very high speed logical operations which makes it useful for systems such as graphics and image processing.

The CE integrates multiple multipliers, adders, registers, magnitude detectors, etc. on a single chip. It has multiple high-performance, parallel inputs and parallel outputs to maintain data-flow from external parallel devices to the on-chip processor. All the on-chip resources could be kept busy for the applications mentioned earlier. When all the computing resources are combined together, the device achieves the computation rate close to one Giga Operations Per Second (GOPS) in a pipelined mode. An operation is defined to be a multiplication, an addition or an equivalent operation. The device operates at an I/O data rate close to 5 Gigabits Per Second to maintain data flow to the processor. The key features of the CE are shown in Table 2.0.

2.1 Operand Formats

Primarily, the CE operates on complex values. Therefore, all the values on the CE are described in a complex format. The simplest way of handling real numbers on the CE is by setting imaginary parts of

complex values to zero. When imaginary parts are also used as real values, extreme care should be taken. The CE uses fractional, two's complement format throughout for the arithmetic operations. For logical operations, the real and imaginary parts of a complex value contain independent logical values. Following are the formats used in the architecture of the CE for the representation of various operands:

a. Single-Precision Complex Values: In this format, a complex value is represented by two independent 16-bit values. One 16-bit value represents the real part and the other represents the imaginary part. For the arithmetic operations, each 16-bit value uses the two's complement format, the most significant bit (msb) being the sign bit. For the logical operations, the real and imaginary parts of a complex value consists of two independent 16-bit logical values.

b. Extended-Precision Complex Numbers: This format is the same as the single-precision format except that the real and imaginary parts are represented by 20 bits instead of 16 bits. It also uses two's complement, fractional format. The format is also referred to as 20-bit complex number format. The 20 bit logical values are not defined for the CE. Intermediated values in this format are produced during the arithmetic computation.

c. Block Floating Point Format: The hardware has been included to provide the block floating-point capability to the CE for FFT applications. The machine has magnitude-detectors at the output stage to monitor the magnitude of each number passing through the output. The detectors can be programmed to monitor the magnitude of a block of N complex numbers passing through the output. At the end of the pass, the machine produces a scale-factor which should be applied to right-shift (scale) every complex-number in the processed block of data before that data-block enters the next FFT computation-pass. The right shift will prevent the machine from overflowing during the pass. Of course, hardware has been included to shift right every data point at the input stage of the CE. The mechanism has been designed to prevent overflow during the FFT algorithm

KEY FEATURES OF CE

Computational Rate up to one GIGA-OPERATIONS per second (1 GOPS).
The Data I/O Rate up to 5 Giga Bits Per Second.
A total of 16 functions (instructions) available on single-chip.
The FFT-specific and general-purpose functions, executed every 80 ns.
A single-chip integration of vector-functions such as FFT Radix-4 butterfly, block multiply, block-add, block-logical functions, etc.
Every 80 ns, operates upon 4 complex-data values and 4 complex auxiliary-data values, and produces 4 complex data values.
For example: 1024 complex-point FFT executed in about 20 μs.

Table 2.0

computation. The machine uses 3 bit, unsigned, binary integer format to represent the Scale Factor. The details of the block floating point scheme will be presented later.

2.2 BASIC ARCHITECTURE

A block diagram of the CE, emphasizing data inputs, auxiliary data inputs, data outputs and pipeline stages is shown in Figure 2.0. As shown in the Figure, all the buses are 16 bits wide to directly interface to parallel memory devices. The processors of the CE may be programmed to implement several different functions as described later. Basically, the processor has been designed to operate upon two sets of four complex values, each set consisting of four 16 bit complex values. After a pipeline latency, the processor produces a set of four 16 bit complex values as an output. As shown in the Figure 2.0, the latency from the input to the output is four machine cycles due to the pipeline register stages RO, R1, R2, and R6. Effectively, these registers are clocked at the machine cycle rate. The machine cycle time is defined to be an interval during which all four complex operands are fed to the device. The machine cycle time is specified to be 80 ns for the CE. Subsequently, the machine cycle time will be referred to an Tm nanoseconds. When the pipeline is running, the processor accepts two sets of four complex numbers every

machine cycle and produces one set of four complex numbers every machine cycle at the output stage. Several different input/output configurations are possible on the CE offering different system trade-offs which will be described below.

As shown in Fig. 2.0, four 16 bit input complex data operands are switched into a set of eight 16-bit registers R0[7:0](15:0), from external input D buses during a machine cycle time Tm. Note that R0[7:0]-(15:0) stands for a set of eight 16 bit registers. The individual registers, in-turn are called R01(15:0), R02-(15:0), etc. During the same machine-cycle, a corresponding input set of four 16-bit complex auxiliary data operands are switched into another set of eight 16 bit registers called RX0[7:0](15:0), from external input AXD buses. On the next machine cycle, the values from set of R0 registers and set of RX0 registers are transferred into set of registers R1 and set of registers RX1 respectively, relieving the input registers to collect the next sets of values. During this machine cycle, several operations on data contained by R1[7:0]-(15:0) are performed by the Processor A and results are latched into registers R2[7:0](15:0). Meanwhile, auxiliary data contained by RX1[7:0](15:0) advances to RX2[7:0](15:0) to maintain alignment with the data. In the next machine cycle, operations on values contained by register sets R2 and register sets RX2 are performed by the Processor B. The Processor B produces additional latency of one machine cycle due to internal pipelining. This latency is shown by including an additional stage of registers in the Processor B in Fig. 2.0. The results (a set of four 16 bit complex values) are latched into a set of output registers R6[7:0]-(15:0) on the following machine-cycle. The values are fed to the output data buses during this machine-cycle over a period of Tm. Therefore, latency of four machine cycles results from the input to the output of the CE.

2.3 INPUTS/OUTPUTS - BM PIN

As described above, computationally the CE operates at a machine-cycle time of Tm ns. However, faster clocks are required to input and output all the values within a machine-cycle. There are two basic modes of input/output bus operations, controlled by the pin BM (Bus Mode).

2.3.1 Dual I/O Mode- DIR Pin and CONFIG Pin

When input pin BM is in a logic low state, the device is said to be in the "Dual I/O" configuration. In the dual I/O configuration, all the input and output operations are done over respective pairs of buses as shown in Figure 2.1. The buses D5, D6, D7, D8, AXD3 and AXD4 are in Don't Care state in this mode. In addition, control over the directionality of the buses is provided via the pin DIR. When pin DIR is in a logic low state, the pair D1 and D2 acts as input buses while pair D3 and D4 acts as output buses, as shown in Figure 2.2. The device is said to be in "right direction mode" in this state. The directionality of the data buses is reversed when DIR pin is set to a logic high state as shown in Figure 2.2, putting the device in the "left direction mode".

Basically, a set of four complex numbers (eight 16-bit values) is transferred over each pair of buses in the dual I/O mode every Tm ns. Therefore, each bus operates at a rate of 4/Tm MHz, transferring a 16-bit word every Tm/4 ns. A CLKIN signal of 4/Tm MHz is needed in the dual I/O mode to handle bus transfers. Every Tm/4 ns, each pair of buses carries corresponding real and imaginary parts of a complex number.

Additional flexibility in the dual I/O mode is provided by an input pin, called CONFIG. In the dual I/O mode discussions, so far, the CONFIG pin was assumed to be in the low state. The pin CONFIG (Configuration) allows the CE to be configured for Dual-Memory Systems (such as shown in Fig 1.3 and Fig 1.4) or for Single-Memory Systems (such as shown in Fig 1.5 and 1.6). When CONFIG pin is low, the CE works at full speed (machine cycle time Tm), continuously reading data from read memories via input buses and continuously writing data to write memories via output buses. As discussed above, when a single external memory is used, it takes twice the memory cycles in a given pass, because the read and write operations are done sequentially rather than in parallel. The CE has to work at half the speed in this case (machine cycle time of 2Tm ns) to allow read and write sequentially. When CONFIG pin is set at a logic high level, the CLKIN signal is internally divided by a factor of two to slow down the CE. In this case, either the bus pair D1/D2 or the bus pair D3/D4, is used for both the read and write operations, as shown in Fig 2.3. The bus pair is dynamically switched by the external DIR signal for alternate read and write operations. The timing will be further clarified in the following sections.

2.3.2 QUAD I/O MODE

The external buses can be reconfigured by putting a logic high signal on the BM pin. In this mode, which is called the quad mode, the CE bus configuration appears as shown in Figure 2.4. Each set of four complex numbers is transferred over corresponding set of four buses instead of two buses in the dual I/O mode. The buses D1, D2, D5 and D6 carry the four input complex-data values and the buses AXD1, AXD2, AXD3 and AXD4 carry the four auxiliary input complex-data values. The buses D3, D4, D7 and D8 carry the four output complex-data values. The bus-direction pin DIR and the CONFIG pin are in the Don't Care mode in this case. Therefore, the directionality of the buses can not be changed in the quad I/O mode.

Since twice the number of buses are available to transfer each set of four complex numbers, each bus operates at half the rate compared to the dual I/O mode. The buses operate at a rate of 25 MHz, transferring a 16 bit value every Tm/2 ns in the quad I/O mode. For each set, real parts of four complex numbers are transferred over corresponding four buses in the first half of the machine cycle. The four corresponding imaginary parts are transferred in the later part of the machine cycle, completing a transfer of four complex numbers.

2.4 THE CLOCK SIGNALS

A model of the internal clock generator of the CE is shown in Fig 2.5. The clock generator is activated by a CLKIN signal from an external pin. The CLKIN signal is fed with a 4/Tm MHz clock or a 2/Tm MHz clock, depending upon the input BM signal. If the device is in a quad I/O mode, the input frequency must be 4/Tm MHz. The input clock rate must be 2/Tm MHz, if the device is in a quad I/O mode. The clock generator is further controlled by the input CONFIG signal. In the dual I/O mode, when CONFIG input is zero, the clock generator generates internal clocks which are called CK2, CK4, CK8, CK8PO, CK8P1, CK8P2 and CK8P3 as shown in Fig 2.6. An external signal, SYNC, is used to synchronize the internal CE clocks with internal clocks of other external devices which share the same SYNC and CLKIN signals. The CLKIN is fed at a frequency of 4/Tm MHz, which produces the CK4 signal at 2/Tm MHz and the CK8 signal at 1/Tm MHz. The CK8P0 to CK8P3 are four non-overlapping quarter phases of the CK8 signal. The signal CK8 appears at the CLKOUT output pin of the CE. When the SYNC input is applied (level high), the clock generator goes into a reset mode, taking various clock outputs to a low level as shown in Fig 2.6. When the SYNC signal makes a transition from high to low, the CK4 and CK8 signals make a transition to logic high on the first following positive transition of CK2.

When CONFIG is set to a high level, the clock generator produces clock waveforms as shown in Fig 2.7. Although, the CLKIN signal is at a frequency of 4/Tm MHz, all the internal waveforms are slowed down by a factor of two due to an internal division of the CLKIN signal by two. The SYNC signal maintains synchronization with the CLKIN signal as described before.

In the quad I/O mode, the clock waveforms appear as shown in Fig 2.8. The CK4, CK8P0, CK8P1, CK8P2 and CK8P3 are in the Don't care state. The SYNC signal operates, as before, in synchronism with the CLKIN signal which is fed at a frequency of 2/Tm MHz.

Note that the logic shown in the Fig 2.5 is a model of the clock generator. The actual implementation on the chip is different. On the chip, minimum skew is achieved between CLKIN and various internal clocks. The clock signals shown in this section will be used throughout these specifications in order to clarify the timing of various elements.

2.5 DEFINITION OF I/O OPERANDS

As stated earlier, the device operates on a set of four single precision, complex data values and a set of four single precision, complex auxiliary data values, every machine cycle. It produces a set of four single precision, complex data values every machine cycle as an output.

Let us define a set of input data operands to be as following.

Data Input Operand O: r0 + j i0

Data Input Operand 1: r1 + j i1

Data Input Operand 2: r2 + j i2

Data Input Operand 3: r3 + j i3

where r# and i# are respectively the real and the imaginary parts of input single-precision complex numbers. The j is the square root of the negative unity. The set of four input data operands will be referred

to as (r + ji)[3:0] in the future.
Similarly, (c + ji)[3:0] is a set of four input auxiliary data operands as following.
Auxiliary Data Input Operand 0: c0 + j s0
Auxiliary Data Input operand 1: c1 + j s1
Auxiliary Data Input operand 2: c2 + j s2
Auxiliary Data Input operand 3: c3 + j s3
In the same way, (x + jy)[3:0] is defined to be a set of single precision, complex, output data values produced by the CE as following.
Data Output Value 0: x0 + j y0
Data Output Value 1: x1 + j y1
Data Output Value 2: x2 + j y2
Data Output Value 3: x3 + j y3
The symbols defined in this section will be used throughout these specifications to refer to input/output values.

Note that when logical operations are performed, then, r0, i0, a0, b0, x0, y0, etc. are treated as independent logical values. However, the term 'complex value' or 'complex number' is used to refer to those operands.

2.5.1 Input/Output - Timing

A timing diagram, showing the timing of various input/output operands, which have been defined in the previous section, is shown in Fig 2.9 for the dual I/O mode when CONFIG pin is held in a low state. The buses are operated four times per machine cycle as mentioned before. Note that, for example, the value (r0 + j i0) is fed in first and the value (r3 + j i3) is fed in at the end of a machine cycle. The various I/O time slots (within a machine cycle) on various buses are referred to as "ro slot", "io slot", "ao slot", etc.

The timing of I/O operands for the dual I/O mode, when CONFIG pin is held in a high state, is shown in Fig 2.10. Note that the CLKIN is at a frequency of 4/Tm MHz and the machine cycle is stretched to Tm/2 ns. Alternate read and write operations are performed over the bidirectional bus pair either D1/D2, or D3/D4. The bus directionality is controlled by an external signal DIR as shown in the Figure 2.10. When DIR signal is applied in accordance with the timing relationships shown, the CE is responsible for providing correct output data and clocking in the correct data. Note that the Figure 2.10 shows the timing of data buses D1 and D2. Instead, if buses D3 and D4 are used, the polarity of the applied signal DIR must be reversed.

The I/O timing for the quad I/O mode is shown in Fig. 2.11. As mentioned above, the device uses all the buses in this mode and each bus is operated at a rate of 2/Tm MHz as shown in Fig 2.11. Note that four real parts of a set of complex numbers are transferred during the first half of the machine cycle. The corresponding four imaginary parts are transferred during the second half of the machine cycle.

2.6 INPUT/OUTPUT - ARCHITECTURE

The I/O architecture of the CE is shown in the block diagram of Fig 2.12.

The device has a set of 8 input/output data buses D[8:1](15:0) and a set of 4 input auxiliary data buses AXD[4:1](15:0) as discussed previously. The buses could be used in the dual or quad I/O mode.

2.6.1 Data Input Section

Internally, the device has a set of four input buses which are called IB[4:1](15:0). As discussed in the previous sections, the device receives a set of four input data values (r + ji)[3:0] every machine-cycle over external data buses. The values are received over internal bus pair IB1/IB2 or over four buses IB1bl/IB2/IB3/IB4, depending upon the BM pin. All together, eight values are clocked in (four real and four imaginary). The in-coming data is channeled through input multiplexers to a set of eight 16 bit registers which are called RO[7:0](15:0). It is assumed that every register, shown in the architecture of CE, is clocked on a positive edge.

Each in-coming data value (r0, i0, r2, i2, etc.) is assigned to one of the RO# registers. The pattern of assignment of various members of the input set of data values (r + ji)[3:0] to various members of the set of registers R0[7:0] may change from instruction to instruction. An instruction is a group of logical values,

applied on six input pins INS(5:0), as shown in control block of Fig 2.12. An instruction configures the Processor A and Processor B to execute various functions described earlier (Butterfly, Block-Multiply, etc). The instruction-set is summarized below and a detailed account of each instruction is presented. Each instruction requires the input set of data (r+ji)[3:0] in a different sequence in the input registers of the Processor A. Therefore, the assignment of input data values to register set R0 changes from instruction to instruction as stated earlier. This is achieved by controlling the input multiplexers and the sequence of clocks to various members of register set R0[7:0] as the input data comes in. The R0[7:0] registers are fed with the clock-set CK8P(3:0) or by CK8/CK8- depending upon the BM mode. All the eight input data values are clocked into R0[7:0] register-set within a machine cycle in a sequence defined by the instruction being executed. The data is clocked into the next set of registers R1[7:0](15:0) by CK8 in the next machine cycle, making it available for the Processor A. The Processor A executes on the data, depending upon the instruction, and results are latched into a set of registers R2[7:0](15:0) which are further processed by the Processor B.

2.6.2 Auxiliary Input Section

In parallel with data, a set of four input auxiliary data values, (c+js)[3:0], is collected over a set of internal auxiliary input buses, XIB[3:1](15:0). The bus pair XIB1/XIB2 or all the four buses XIB[3:1] may be used depending upon the I/O modes. As in the case of data, the auxiliary data is clocked into a set of input auxiliary registers RX0[7:0](15:0) using the multiplexers and controlling the clock-inputs to the registers. Again, the assignment of input auxiliary values to the registers RX0[7:0] may vary from instruction to instruction. All the eight values, (c+js)[3:0], are clocked into the register set RX0[7:0] over a machine cycle in a sequence defined by the instruction. In the next machine cycle, the values are moved to the register set RX1[7:0](15:0) by CK8 signal. The auxiliary data is further delayed by a machine cycle by moving it into the next set of registers, RX2[7:0](15:0), by CK8 signal. A series of 'one bit right shift and round' circuits have been included in between each member of the register-set RX1[7:0] and the corresponding member of register-set RX2[7:0]. Each input auxiliary value may be shifted by one bit to the right by controlling the XSIN pin. After shifting, a Round bit, RNDX, is added to the lsb of the shifted value to perform unbiased rounding. The RNDX is determined by the following logical equation.

RNDX = DB1 . DB0

where DB0 is the discarded lsb bit and DB1 is the next higher significant bit. Note that when logical instructions are implemented, the shift is logical and no rounding is performed.

When auxiliary data is clocked into register-set RX2[7:0], it gets aligned with the data contained by the register set R2[7:0]. Both register-sets feed the data to the Processor B for further computation.

2.6.3 Data Output Section

The Processor B, which causes an additional latency of one machine-cycle due to its internal pipelining, produces a set of values (x+jy)[3:0] in the register set R6[7:0](15:0). The processor B assigns various members of the set (x+jy)[3:0] to various registers in the set R6[7:0], in different ways for different instructions.

After the output values have been produced in the register set R6[7:0], those are read to the internal output bus set OB[4:1](15:0) during the same machine cycle. Either the bus pair OB1/OB2, or all the buses OB1/OB2/OB3/OB4 are used, depending upon the I/O modes. The output values from the register-set, R6-[7:0] are sequenced to the output buses by using tri-state buffers and multiplexers at the output of registers, in accordance with the timing discussed above. Either the clocks CK8P[3:0], or CK8/CK8- are used to drive the tri-state buffers, depending upon the I/O mode. The assignment and timing of the output registers will be further clarified in the instruction description.

2.7 INSTRUCTION-SET; OPCODES AND SUMMARY

The CE has a variety of programmable input/output options which have been described so far. In addition, the Processor A and Processor B may be programmed to execute a variety of instructions as shown in Table 2.2.

MNEMONIC	DESCRIPTION
DEDICATED FFT CLASS	
BFLY4	A Radix-4, DIF, FFT Butterfly.
BFLY 2	Two Radix-2, DIF, FFT Butterflies
FFT2N	Recombine N Complex Point FFT to 2N Real Point FFT
FFTNN	Recombine N Complex Point FFT to two N Real Point FFTs.
GENERAL ARITHMETIC CLASS	
AFLOW	Arithmetic Flow through: Pass data Complemented/Unaffected.
BMUL	Block multiply two sets of complex numbers.
BSQSM	Block Square/Sum a set of complex numbers.
BADD	Block Add two sets of complex numbers
BSUB	Block Subtract two sets of complex numbers
BCONJ	Block Conjugate a set of complex numbers
BCONS	Block of Constants: produce ZERO or ONE on outputs.
BCMUL	Block Multiply two sets of Complex Values with one set Conjugated.
GENERAL LOGICAL Class	
LFLOW	Logical Flow through: Pass Data Inverted or unaffected.
BAND	Block AND two sets of complex values
BOR	Block Or two sets of complex values
BXOR	Block XOR two sets of complex values.

Table 2.2: Instruction Summary

As described in the Table 2.2, there are three classes of instructions. There is a Dedicated FFT Class which contains four instructions, dedicated to the specific FFT algorithms. There are eight instructions in the General Arithmetic Class. These instructions are fairly general purpose in nature. Some of the instructions such as BMUL, BSQSM and BADD are also useful for FFT systems. The BMUL is useful for FFT windowing and BSQSM can be used for post-FFT squaring/summing. The BADD can be used for post-FFT averaging, etc. All these instructions can be used to implement more general purpose arithmetic algorithms. The General Logical Class of instructions contains four instructions to perform various logical operations. Although the CE has been primarily designed for arithmetic intensive applications, the logical capability makes it useful where very high performance bit manipulation may be desired in addition to arithmetic. One such application is image/graphics processing.

Basically, all the instructions operate upon one set or two sets of four complex values and produce one set of complex values. The instruction bits I(5:0) (Table 2.2) are defined at the input pins of the device. The Primary Function of an instruction is defined by the bits I(3:0). The bit I4 is called CI which stands for Conjugate-Input or Complement-Input. The bit CI, which is active high, is used to conjugate, arithmetically-complement or logically-complement the incoming data values. The functionality of the CI depends upon the Class of instructions. In FFT Class, the CI is used for conjugate control. All the incoming complex data values are conjugated by two's complementing the imaginary parts before operations are performed by the instruction. In the Arithmetic Class, the CI acts as a global two's complement control. All the incoming complex values (real and imaginary) are two's complemented before the values are operated upon. The CI also acts as complement control for the logical Class of instructions. However, in this case, all the incoming values are logically complemented (inverted) before the operation, defined by the instruction, is performed.

The CO bit behaves in a similar fashion for three classes of instructions, except that conjugate or complement operation is performed at the output. First the operation, defined by the instruction, is performed. Then, the outgoing data is conjugated or complemented. Note that the CI impacts the incoming data through the data-input section only. It does not impact the auxiliary input data.

The conjugate control, along with FFT instructions, can be used to implement inverse FFTs. The

conjugate facility also allows the implementation of time-domain filtering by using frequency domain techniques. The complementing facilities, along with the instructions in the Arithmetic Class and the Logical Class, provides additional functionality for a variety of applications.

An instruction much be set-up one machine-cycle in advance of the data. Typically, the instruction will remain static for processing a block of N complex values.

2.8 PROCESSOR-A ARCHITECTURE

The processor A derives its inputs from the register set R2[7.0]. The processor A is basically dedicated to perform initial summing operations associated with various FFT oriented instructions (BFLY4, BFLY2, FFT2N, etc) which are shown in the instruction description. For general purpose instructions (such as BMUL, BADD, BXOR), the processor A is basically in the flow-through mode to pass the data to the processor B.

2.8.1 Input Complementers

A block diagram of the processor A is shown in Fig 2.13. The eight inputs to the processor A are first passed through a set of two's complementers, designated as COMO-COM7 in Figure 2.13. The complementers are directly controlled by the input pins COMP(7:0) or by the I4 bit in the instruction, depending upon the SPLM (SPECIAL MODE) pin and the class of effective instruction. When SPLM pin is in a high state each Complementer is directly controlled by the values on pins COMP(7:0), for FFT class of instructions. For example, if COMPO is high, then complementer COMO, produces a two's complement when data passes through it. If COMPO is low then data passing through COMO is unaffected. Similarly, complementers COM1-COM7 are affected by the values on pins COMP1-COMP7. When pin SPLM is in a low state, the complementers are not affected by inputs COMP(7:0). Instead the bit I4 (Conjugate Input - CI) controls the input complementers. When I4 is low, the data through Complementers is unaffected. If I4 is high, the data values are affected as described in the descriptions of individual instructions.

2.8.2 Shift And Round

After complementers, the data is passed through a set of Shift and Round circuits, designated as SRO-SR7 in Fig 2.13. The SR blocks produce a shift on incoming data as defined by pins SFI(2:0). All the SR blocks produce the same amount of shift. The sign bit is extended into the most significant bit (msb) positions if arithmetic instruction is being implemented, otherwise, zeros are inserted in the msb positions. Table 2.3 describes the SFI(3:0) inputs and the produced shift.

TABLE 2.3

Shift Values			
SFI2	SFI1	SFI0	PRODUCED SHIFT
0	0	0	NO SHIFT
0	0	1	1 BIT
0	1	0	2 BITS
0	1	1	3 BITS
1	0	0	4 BITS
OTHER STATES			NOT VALID

As shown in the Table 2.3, shift up to four only is allowed. When arithmetic instructions are implemented, the shifted values are rounded to perform an Unbiased Round, by adding a bit RND to the shifted value. The bit RND is determined by the following logical equation.

RND = (SFI2-. SFI1 -. SFI0 . DB1 . DB0) +
(SF2- . SF1 . SF0- . DB1 . [DBO + DB2]) +

(SF2- . SF1 . SFO . DB2 [DBO + DB1 + DB3]) +
(SF2 . SF1- . SF0- . DB3 . [DBO + DB1 + DB2 + DB4])

Note that DBO to DB4 are unshifted data bits; the bit DBO being the least significant bit (lsb). The RND bit is forced to be a zero when logical instructions are implemented.

2.8.3 Sum Stages

The data passes through a first set of adder stages, designated as PS0 to PS7 (Programmable Summers) in Fig 2.13. The values are further combined by a second stage of Programmable Summers, called PS8-PS15. A Programmable Summer is an adder, containing programmable inputs as shown in the notation of Fig 2.13. Either input of a PS can be two's complemented. Either or both inputs may also be programmed to be zero, disregarding the incoming data. In this way, a PS can be used to add, subtract, flow-through or produce zeros. The PS are automatically programmed by the instruction bits I(3:0) as described in the instruction descriptions. The output from the second PS stage is directly latched into the R2[7:0] register-set. The interconnections of PS devices have been determined to implement various FFT instructions. However, data flows through unaffected for other instructions.

2.8.4 Definition of Intermediate Operands

For arithmetic instructions, typically, the Processor A produces a set of four single-precision, complex values. Let us define those values to be a set (m + jm)[3:0]. The value (m0 + jn0) is contained by the register-pair R20 and R21; m0 being contained by register R20, and n0 being contained by R21. Similarly, the remaining values are contained by the remaining registers in the same sequence.

2.9 PROCESSOR B: ARCHITECTURE

The Processor B operates upon two sets of complex numbers, a set (m + jn)[3:0] contained by R2[7:0] registers, and a set (c + js) [3:0] contained by RX2[3:0] registers. A block diagram of the Processor B is shown in Fig 2.14. The Processor B has been defined to work at twice the speed of Processor A, operating twice in a machine cycle. The Processor B contains eight 16 x16 multipliers, four 20 bit ALUs and several pipeline registers. The internal pipeline registers of the Processor B are clocked by the 2/Tm MHz clock, CK4, due to the processor cycle-time of Tm/2 ns. Basically, the processor is organized to perform four complex multiplications in one machine cycle. The two multipliers and one ALU is interconnected to form one 'Arm' of the Processor, as shown in Fig 2.14. All the four Arms shown in Fig 2.14 are identical except the highlighted data paths. The highlighted data paths allow ALU0 to be able to feed the data to ALU1. Similarly, the ALU2 can send the data to the ALU3. Those 'special' paths are used by only one instruction, FFT2N. In the normal mode (when all other instructions are executed), the four Arms appear identical as seen by ignoring highlighted data paths in Fig 2.14. For those instructions, all the Arms are identical and independent. This is called 'normal' mode of operation. When, FFT2N instruction is executed, it uses the 'special' mode of operation.

Referring again to Fig. 2.13 and Fig. 2.14, dashed lines indicate the flow of operands through Processor A and Processor B for the radix-4, decimation-in-frequency, fast Fourier transform, using the butterfly algorithm (BFLY4 in Table 2.2). The illustrated flow paths are the same for both phases of the algorithm implementation. The input and output paths for the operands and for the algorithm constants pass through the apparatus shown in Fig. 2.11, only the output path varying between the phases. The similar operand flow diagrams can be provided for each of the algorithms described in Table 2.2.

2.9.1 Processor B: Normal Mode

In the Normal Mode, each Arm (Fig. 2.14) has data paths to execute a complex multiply operation in one machine-cycle. For example, the Arm 0 operates on two input complex-numbers, (m0 + jn0) and, say, (c0 + js0). Both input numbers remain stable for a complete machine-cycle, Tm ns, in respective registers R2# and RX2#. During the first half of Tm, the multiplexers of MULO and MUL1 are configured to produce m0.c0 and n0.s0, respectively. The products are left justified, truncated to 20 bits, and latched into registers

R30 and R31 which are clocked by a CK4 clock. Note that the registers R3= are implemented inside the multiplier as intermediate pipeline registers. For the sake of convenience, those are shown at the outputs of the multipliers in Fig 2.14. Next, the products are transferred to registers R40 and R41, which are clocked by CK4. Then, the ALU0 performs a function (m0.c0 - no.s0) which is the real part of the complex multiplication. The computed value is rounded to 16 bits, then passed through a two's complementer COMP0, and, then, latched into an intermediate register R50. After producing m0.c0 and n0.s0, the multiplexers of MUL0 and MUL1 are switched in the second half of the machine cycle. In the second half, MUL0 and MUL1 produce n0.c0 and m0.s0, respectively. As before, the data flows through the CK4 pipeline, following the previous data. The ALU0 performs a function (n0.c0 + m0.s0), which is an imaginary part of the output. The result is latched into register R61 by a CK8 signal. At the same time, the corresponding output real-term, residing in the register R50, is latched into register R60. Therefore, registers R60 and R6(1) contain results of a complex multiplication, which have been termed (x0 + jy0) previously, as an output. The timing of the Arm is further clarified in the timing diagram of Fig 2.15. Note that the registers R3# and R4# produce an additional latency of one machine-cycle. This latency has been shown by including a stage of artificial CK8 pipeline registers in the equivalent diagram of the Processor B in Fig 2.0.

As shown, the multiplexers of MUL0 and MUL1 (Fig 2.14) have provisions to produce m0**2 and n0**2, which can be further added by ALU0 to execute a SQSUM instruction. The input data can also be passed through the multipliers unaffected by selecting a'1' on the multiplexers. The data then can be operated upon by using the ALU. The ALU supports various functions such as Add with either input complemented, Logical AND, Logical OR, Logical EXOR, One's Complement, as required by various instructions.

In the Normal Mode, all the four Arms operate as described above. The Processor B operates in a different mode, called Paired Mode, when the instruction FFT2N is executed, which is described in the next section.

2.9.2 Processor B: Paired-Mode

In the Paired-Mode, Arm-0 and Arm-1 are interconnected together by using the highlighted path, as shown in Fig 2.14. Similarly, Arm-2 and Arm-3 are interconnected. These data paths are used by the instruction FFT2N as described in its description in the next chapter. In this mode, only one complex-value is produced as an output, by Arm-1 and Arm-3.

The mathematical equations, implemented for FFT2N, were discussed in the instruction set description. It can be easily verified that the latency from the input to the output of the Processor B in the Paired Mode is the same as the Normal Mode (One Machine-Cycle), by producing a timing diagram similar to that shown in Fig 2.15.

2.9.3 Processor B: Round And Complement

When ALU0-ALU3 implement arithmetic instructions (Fig 2.14), their outputs are rounded to 16 bits. The output of each ALU is first truncated to 16 bits and, then, a bit RNDB is added. The RNDB is determined by the following logical equation to perform unbiased rounding.

RNDB = DB2 . [DB0 + DB1 + DB3]

Note that DB0 - DB3 are the discarded bits on truncation, the DB0 being the lsb.

The data is passed through the Complementers after rounding as shown in Fig 2.14. The effect of complementers on the output data is described in the individual description of each instruction.

2.12 VARIOUS SIGNALS AND PIPELINING

The CE is a pipelined device as mentioned earlier. All the input/output signals have been carefully pipelined so that the pipelining does not cause any inconvenience to the user. The data pipelining have been well explained above. The pipelining of various signals is described in relation to the input/output data.

Various I/O signals of the CE can be divided into separate categories, as discussed below.

2.12.1 General Pipelined Signals

These signals flow in a pipeline along with the data to control various blocks of the CE. These signals should be applied, at least, one machine cycle before the machine cycle during which the first set of data-block is inputted (to be impacted). Following are the signals in this category.

SFI(2:0) - Scale Factor Inputs
XSIN - Auxiliary Scale Factor Input
I(5:0) - Instruction Inputs
SPLM - Special Mode Bit
RMB - Radix Mode Bit
EOP - End of Pass Signal
BOP - Beginning of Pass Signal

All these signals are sampled into CE on a positive-going edge of the 1/Tm(CK8) clock. Due to the internal pipelining of these signals, to match the data pipeline, the data can be applied in the next machine cycle, without having to wait for the propagation of those signals to all the pipeline stages. These signals can be changed to start a different pass, while the data corresponding to a previous pass is still flushing through the pipe. Although, in typical applications, these signals will be changed after the data for a current pass is flushed through the chip.

All the signals, except EOP and BOP, are static signals, which typically remain static on pins for the whole pass. Therefore, these signals are called Pass-Static. The BOP and EOP are pulsed signals which are activated only once during each Pass. These are called Pass-Pulsed signals. The EOP and BOP are sampled on the positive edge of the 1/Tm clock. The EOP and BOP should not exceed the machine-period of the CE.

3.0 INSTRUCTION-SET

Every instruction will now be described. The input and output aspects of the CE have been described earlier indicating the programmable options available to the user. These I/O programming options apply to all the instructions as ways of inputting/outputting data values.

When describing each instruction, it will be assumed that I/O data buses have been already set-up as desired, depending upon the external system configuration. The following parameters will be used to make reference to the I/O data values which have been defined previously:

Data Input Values:
r0 + j io, rl + j il, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

These parameters also have timing information associated as well. The timing slots during which various I/O values are transferred as functions of Bus-control signals has been described. In the description of various instructions, the Output Values will be shown as functions of the Input Data Values and Input Auxiliary Data Values. Of course, it should be understood that the Output Values are produced after a latency as described earlier. Various control signals which impact a given instruction will also be described. The impact of a given instruction on any output control signal will also be mentioned.

3.1 INSTRUCTION BFLY4

Description: It implements a mathematical function corresponding to Decimation-in-frequency, radix-4 butterfly.

Data Input Values:
r0 + j i0, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

First, intermediate results are produced which are defined as following:
a0 + j b0 = [(r0+r2) + (r1+r3)] + j [i0+i2) + (i1+i3)]
a1 + j b1 = [(r0-r2) + (i1-i3)] + j [(i0-i2) -(rl-r3)]
a2 + j b2 = [(r0+r2) - (r1+r3)] + j [(i0+i2) -(i1+i3)]
a3 + j b3 = [(r0-r2) - (i1-i3)] + j [(i0-i2) + (r1-r3)]
Then, the output values are produced as following.
x0 + j y0 = (a0 + j b0) * (c0 + j s0)
x1 + j y1 = (a1 + j b1) * (c1 + j s1)
x2 + j y2 = (a2 + j b2) * (c2 + j s2)
x3 + j y3 = (a3 + j b3) * (c3 + j s3)
The above mentioned function collapses to a standard Radix-4, DIF butterfly when c0 = 1 and s0 = 0. Therefore, it Radix-4 Butterfly is to be computed, the user must ensure that incoming data c0 and s0 have those values.

3.2 INSTRUCTION BFLY2

DESCRIPTION: The instruction executes two FFT, Radix-2, Decimation-in-Frequency butterflies.
Data Input Values:
r0 + j io, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

First, intermediate results are produced, defined as following.
a0 + j b0 = (r0+r1) + j (i0-i1)
a1 + j b1 = (r0-r1) + j (i0-i1)
a2 + j b2 = (r2+r3) + j (i2+i3)
a3 + j b3 = (r2-r3) + j (i2-i3)
Then, output values are produced as following.
x0 + j y0 = (a0 + j b0) * (c0 + j s0)
x1 + j y1 = (a1 + j b1) * (c1 + j s1)
x2 + j y2 = (a2 + j b2) * (c2 + j s2)
x3 + j y3 = (a3 + j b3) * (c3 + j s3)
Note that If CO = C2 = 1, and SO = S2 = 0, then the above equations conform to two separate radix-2 butterflies. The r0+ji0 and r1+ji1 become inputs for the first butterfly. The x0+jy0 and x1+jy1 become corresponding outputs. Similarly, the r2+ji2 and r3+ji3 become inputs for the second butterfly. The x2+jy2 and x3+jy3 become corresponding outputs. The c1+js1 and c3+js3 are twiddle-factors for respective butterflies. The user must ensure above mentioned values for c0, s0, c2 and s2 are supplied.

3.3 INSTRUCTION FFT2N

DESCRIPTION: The instruction recombines a. FFT of N complex-points into a FFT of 2N real-points. (Reference: The Fast Fourier Transform, E. Oran Brigham, Prentice-Hall, 1974, p.167).
Data Input Values:
r0 + j i0, Don't Care, r2 + j i2, Don't Care.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Output Data Values:
x0 + j y0, Don't Care, Don't Care, Don't Care

FUNCTION

x0 = (r0/2 + r2/2) + c0*(i0/2 + i2/2) - s0*(r0/2 - r2/2)
y0 = (i0/2 + i2/2) + s0*(i0/2 + i2/2) - c0*(r0/2 - r2/2)

These equations recombine two points of a N complex-point FFT into one point of 2N real-point FFT. (Reference: The Fast Fourier Transform, E. Oran Brigham, Prentice-Hall, 1974, p 169).

3.4 INSTRUCTION FFTNN

DESCRIPTION: The instruction recombines a FFT of N complex-points into two separate N real-point FFTs. (Reference: The Fast Fourier Transform, E. Oran Brigham, Prentice-Hall, 1974, p. 166).
Data Input Values:
r0 + j io, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = [r0/2 + r1/2] + j [i0/2 - i1/2]
x1 + j y1 = [i0/2 + i1/2] + j [r1/2 - r0/2]
x2 + j y2 = [r2/2 + r3/2] + j [i2/2 - i3/2]
x3 + j y3 = [i2/2 + i3/2] + j [r3/2 - r2/2]
These equations recombine four points of N complex-point FFT into two points of N real-point FFT of function-1 and two points of a N real-point FFT of function-2. (Reference: The Fast Fourier Transform, E. Oran Brigham, Prentice-Hall, 1974, p. 167).

3.5 INSTRUCTION AFLOW

DESCRIPTION: The instruction passes a set of four complex values from the input to the output of the device. If desired, values may be complemented (negated) during the Pass.
Data Input Values:
r0 + j i0, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = r0 + j i0, if CO = 0
= -(r0 + j i0), if CO = 1
x1 + j y1 = r1 + j i1, if CO = 0
= -(r1 + j i1), if CO = 1
x2 + j y2 = r2 + j i2, if CO = 0
= -(r2 + j i2), if CO = 1
x3 + j y3 = r3 + j i3, if CO = 0
= -(r3 + j i3), if CO = 1

3.6 INSTRUCTION BMUL

DESCRIPTION: The instruction multiplies members of a block of four complex values with corresponding members of another block of four complex values.
Data Input Values:
r0 + j i0, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = (r0 + j i0) * (c0 + j s0)
x1 + j y1 = (r1 + j i1) * (c1 + j s1)
x2 + j y2 = (r2 + j i2) * (c2 + j s2)
x3 + j y3 = (r3 + j i3) * (c3 + j s3)

3.7 INSTRUCTION BSQSM

DESCRIPTION: The instruction operates on eight data values. It squares each data value and, then, adds pairs together. The instruction can be used to compute magnitude-square if input data values are complex. If real, then the instruction can be used to square and sum a data-array by using an off-chip accumulator.

The input values are 16 bit, single precision values. It produces four 20 bit, extended precision, values. The corresponding x and y outputs are combined to represent a 20 bit value. For example, the x0 and y0 are used to represent the first 20 bit output value. The msb 16 bits are represented by the x0 and the 4 lbs bits are represented by the y0 value. The 4 bits are contained by the most significant part of the y0 value. The lower 12 bits of the y0 are invalid. The values (x1, y1), (x2, y2) and (x3 and y3) are generated in the same way.

Data Input Values:
r0, i0; r1, i1; r2, i2; r3, i3.
Auxiliary Input Values:
Don't Care; Don't Care; Don't Care; Don't Care.
Output Data Values:
x0, y0; x1, y1; x2, y2; x3, y3.

3.8 INSTRUCTION BADD

DESCRIPTION: The instruction adds members of a block of four complex values with corresponding members of another block of four complex values.

Data Input Values:
r0 + j i0, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = (r0 + j i0) + (c0 + j s0)
x1 + j y1 = (r1 + j i1) + (c1 + j s1)
x2 + j y2 = (r2 + j i2) + (c2 + j s2)
x3 + j y3 = (r3 + j i3) + (c3 + j s3)

3.9 INSTRUCTION BSUB

DESCRIPTION: The instruction subtracts members of a block of four complex values from corresponding members of another block of four complex values.

Data Input Values:
r0 + j io, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = (r0 + j i0) - (c0 + j s0)
x1 + j y1 = (r1 + j i1) - (c1 + j s1)
x2 + j y2 = (r2 + j i2) - (c2 + j s2)

x3 + j y3 = (r3 + j i3) - (c3 + j s3)

3.10 INSTRUCTION BCONJ

DESCRIPTION: The instruction conjugates a set of four complex values. If desired, values may be further complemented (negated) at the output.
Data Input Values:
r0 + j io, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = r0 - j i0, if CO = 0
= -(r0 - j i0), if CO = 1
x1 + j y1 = r1 - j i1, if CO = 0
= -(r1 - j i1), if CO = 1
x2 + j y2 = r2 - j i2, if CO = 0
= -(r2 - j i2), if CO = 1
x3 + j y3 = r3 = j i3, if CO = 0
= -(r3 - j i3), if CO = 1

3.11 INSTRUCTION BCONS

DESCRIPTION: The instruction produces a block of constants at the output, disregarding the inputs. The values of constants may be zero or one.
Data Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = 0 + j 0, if CO = 0
= 1 + j 1, if CO = 1
x1 + j y1 = 0 + j 0, if CO = 0
= 1 + j 1, if CO = 1
x2 + j y2 = 0 + j 0, if CO = 0
= 1 + j 1, if CO = 1
x3 + j y3 = 0 + j 0, if CO = 0
= 1 + j 1, if CO = 1

3.12 INSTRUCTION BCMUL

DESCRIPTION: The instruction conjugates the members of a block of four complex values and, then, multiplies those members with corresponding members of another block of four complex values.
Data Input Values:
r0 + j i0, r1 + j i1, r2 + j i2, r3 + j i3.
Auxiliary Input Values:
c0 + j s0, c1 + j s1, c2 + j s2, c3 + j s3.
Output Data Values:
x0 + j y0, x1 + j y1, x2 + j y2, x3 + j y3.

FUNCTION

x0 + j y0 = (r0 - j i0) * (c0 + j s0)
x1 + j y1 = (r1 - j i1) * (c1 + j s1)
x2 + j y2 = (r2 - j i2) * (c2 + j s2)
x3 + j y3 = (r3 - j i3) * (c3 + j s3)

3.13 INSTRUCTION LFLOW

DESCRIPTION: The instruction passes a set of four complex values from the input to the output of the device. If desired, values may be logically inverted during the Pass.
Data Input Values:
r0, i0; r1, i1; r2, i2; r3, i3.
Auxiliary Input Values:
Don't Care, Don't Care, Don't Care, Don't Care.
Output Data Values:
x0, y0; x1, y1; x2, y2; x3, y3.

FUNCTION

x0 = r0; y0 = i0; If, CO = 0.
x0 = r0-; y0 = i0-; If, CO = 1.
x1 = r1; y1 = i1; If, CO = 0
x1 = r1-; y1 = i1-; If, CO = 1.
x2 = r2; y2 = i2; If, CO = 0.
x2 = r2-; y2 = i2-; If, CO = 1.
x3 = r3; y3 = i3; If, CO = 0.
x3 = r3-; y3 = i3-; If, CO = 1.

3.14 INSTRUCTION BAND

DESCRIPTION: The instruction logically ANDs the members of a block of four complex values with corresponding members of another block of four complex values.
Data Input Values:
r0, i0; r1, i1; r2, i2; r3, i3.
Auxiliary Input Values:
c0, s0; c1, s1; c2, s2; c3, s3.
Output Data Values:
x0, y0; x1, y1; x2, y2; x3, y3.

FUNCTION

x0 = r0 .&. c0; y0 = i0 .&. s0;
x1 = r1 .&. c1; y1 = i1 .&. s1;
x2 = r2 .&. c2; y2 = i2 .&. s2;
x3 = r3 .&. c3; y3 = i3 .&. s3;

3.15 INSTRUCTION BOR

DESCRIPTION: The instruction logically ORs the members of a block of four complex values with corresponding members of another block of four complex values.
Data Input Values:
r0, i0; r1, i1; r2, i2; r3, i3.
Auxiliary Input Values:
c0, s0; c1, s1; c2, s2; c3, s3.
Output Data Values:
x0, y0; x1, y1; x2, y2; x3, y3.

FUNCTION

x0 = r0 .OR. c0; y0 = i0 .OR. s0;
x1 = r1 .OR. c1; y1 = i1 .OR. s1;
x2 = r2 .OR. c2; y2 = i2 .OR. s2;
x3 = r3 .OR. c3; y3 = i3 .OR. s3;

3.16 INSTRUCTION BXOR

DESCRIPTION: The instruction logically Exclusive-ORs the members of a block of four complex values with corresponding members of another block of four complex values.

Data Input Values:
r0, i0; r1, i1; r2, i2; r3, i3.
Auxiliary Input Values;
c0, s0; c1, s1; c2, s2; c3, s3.
Output Data Values:
x0, y0; x1, y1; x2, y2; x3, y3.

FUNCTION

x0 = r0 .XOR. c0; y0 = i0 .XOR. s0;
x1 = r1 .XOR. c1; y1 = i1 .XOR. s1;
x2 = r2 .XOR. c2; y2 = i2 .XOR. s2;
x3 = r3 .XOR. c3; y3 = i3 .XOR. s3;

The implementation of the CE, which has been discussed in this specification, follows the general architectural concept which is shown in Figure 3.1.

A variation on the architecture which is configured as shown in Figure 3.2, could also be implemented. The detailed implementation of such architecture will be similar to that described in this specification. The processor A and processor B (see Figures 2.12, 2.13 and 2.14) can be interchanged to achieve such implementation. The resulting architecture will be extremely useful for implementing the majority of functions discussed previously. The butterfly implementations will follow the decimation-in-time algorithm which is shown in Figure 3.3.

The foregoing description is included to illustrate the operation of the preferred embodiment and is not meant to limit the scope of the invention. The scope of the invention is to be limited only by the following claims. From the foregoing description, many variations will be apparent to those skilled in the art that would yet be encompassed by the spirit and scope of the invention.

The content and disclosure of all the claims is included expressly by reference into this specification. Independent claims 1, 2, 3 and 16 describe the apparatus according to the invention under different aspects. Fig. 2.13 and Fig. 2.14 (Processor A and Processor B) are reversed in Fig. 2.12 as compared to claim 1. The associated dependent claims characterize particular embodiments and details of such apparatus. Claims 27, 28 FFT and 29 describe the new method of performing FFT as used by the invention.

Claims

1. An integrated circuit for processing an array of digital data, **characterized by**:

a) at least four primary digital inputs (Fig. 3.1 Primary Inputs; Fig. 20.0: D1, D2, D5, D6), each of said primary digital inputs being capable of receiving a plurality of digital bits;

b) at least two auxiliary digital inputs (Fig. 3.1: Auxiliary Inputs; Fig. 2.0: AXD1, AXD2, AXD3, AXD4), each of said auxiliary digital inputs being capable of receiving a plurality of digital bits;

c) first latching means (Fig. 3.1: Latches; Fig. 2.12: Input Section), said first latching means including one data flow path for each of said primary inputs and means for delaying the flow of data in each of said data flow paths in said first latching means;

d) second latching means (Fig. 3.1: Latch; Fig. 2.12 Auxiliary Input Section), said second latching means including one data flow path for each of said auxiliary inputs and means for delaying the flow of data in each of said data flow paths in said second latching means;

e) first means (Fig. 3.1: Adders/Substract; Fig. 2.13: PS0, PS1, PS2, PS3, PS8, PS9, PS10, PS11) for selectively performing at least one addition or subtraction operation between the data on at least a first and second of said data flow paths in said first latching means to provide at least a first result;

f) second means (Fig. 3.1: Adders/Substract; Fig. 2.13: PS4, PS5, PS6, PS7, PS12, PS13, PS14, PS15) for selectively performing at least one addition or subtraction operation between the data on at least a third and a fourth of said data flow paths in said first latching means to provide at least a second result;

g) first means (Fig. 3.1: Multiplier; Fig. 2.14: MUL0-MUL3) for selectively performing at least one multiplication operation with the data on a first of said auxiliary inputs and said first result, to provide at least a first product;

h) second means (Fig. 3.1: Multiplier; Fig. 2.14: MUL4-MUL7) for selectively performing at least one multiplication operation with the data on a second of said auxiliary inputs and said second result, to provide at least a second product;

i) first arithmetic means (Fig. 3.1: ALU; Fig. 2.14: ALU0, ALU1) for performing at least one addition or subtraction operation with said first product;

j) second arithmetic means (Fig. 3.1: ALU; Fig. 2.14: ALU2, ALU3) for performing at least one addition or subtraction operation with said second product; and means for outputting at least a portion of the data from said first or second arithmetic means.

2. An integrated circuit for processing an array of digital data, **characterized by:**

a) at least four primary digital inputs (Fig. 3.2: Primary Inputs), each of said primary digital inputs being capable of receiving a plurality of digital bits;

b) at least two auxiliary digital inputs (Fig. 3.2: Auxiliary Inputs), each of said auxiliary digital inputs being capable of receiving a plurality of digital bits;

c) first latching means (Fig. 3.2: Latches), said first latching means including one data flow path for each of said primary inputs and means for delaying the flow of data in each of said data flow paths in said first latching means;

d) second latching means (Fig. 3.2: Latch), said second latching means including one data flow path for each of said auxiliary inputs and means for delaying the flow of data in each of said data flow paths in said second latching means;

e) first means (Fig. 3.2: Muliplier) for selectively performing at least one multiplication operation with the data on at least a first of said primary inputs and the data on a first of said auxiliary inputs, to provide at least a first product;

f) second means (Fig. 3.2: Multiplier) for selectively performing at least one multiplication operation with the data on at least a second of said primary inputs and the data on a second of said auxiliary inputs, to provide at least a second product;

g) first arithmetic means (Fig. 3.2: ALU) for performing at least one addition or subtraction operation with said first product to provide at least a first result;

h) second arithmetic means (Fig. 3.2: ALU) for performing at least one addition or subtraction operation with said second product to provide at lest a second result;

i) first means (Fig. 3.2: Adders/Subtract) for selectively performing at least one addition or subtraction operation between said first result and said second result;

j) Fig. 3.2: Adders/Subtract) for selectively performing at least one addition or subtraction operation between said first result and said second result; and

k) means (Fig. 3.2; Output Means) for outputting at least a portion of the data from said first and second means for selectively performing at least one addition and subtraction operation between said first result and said second result.

3. Apparatus for processing digital signals, **characterized by:**

a) input register means (Fig. 2.12: Input Section) for receiving input digital signal groups to be processed;

b) first processor means (Fig. 2.12: Processor A shown in Fig. 2.13) responsive to control signals for providing intermediate signal groups from said input signal groups, said intermediate signal group formats determined by said control signals;

c) second processor means (Fig. 2.12: Processor B shown in Fig. 2.14) responsive to said control signals for combining said intermediate signal groups and scaling digital signal groups to form resulting signal groups in a manner determined by said control signals; and

d) output register means (Fig. 2.12: Output Section) for storing said resulting signals, wherein said output register means receives said resulting signals, said first processor means and said second processor means performing a radix-4 butterfly algorithm (Section 3.1) in response to first control signals.

4. The apparatus of Claim 3, **characterized in that** said register means receives input digital signal groups corresponding to four complex numbers, said scaling digital signal groups coresponding to four complex numbers, said resulting signal groups corresponding to four complex numbers.

5. The apparatus of Claim 3 or 4, **characterized in that** said first processor means includes adder/subtracter units (Fig. 2.13: PS0-PS15) with cross-coupling controlled by said control signals (Fig. 2.13: SPLM, SFI, COMP, etc.), said second processor including a plurality of multiplier units (Fig. 2.14: MUL0-MUL8), each multiplie unit multiplying a scaling digital signal group and an associated intermediate signal group, said second processor means further including a plurality of arithmetic logic units (Fig. 2.14: ALU0-ALU4).

6. The apparatus of one of the claims 3 to 5, **characterized in that** said first processor means and said second processor means perform a radix-2 algorithm (Section 3.2 Instruction BFLY2) in response to second control signals, wherein said input signals correspond to sets of two complex numbers, said resulting signals corresponding to two sets of two complex numbers.

7. The apparatus of Claim 6, **characterized in that** said first processor means and said second processor means recombine a fast Fourier transform (Section 3.4 Instruction FFTNN) of N complex points into a fast Fourier transform of 2N real points in response to third control signals, said first processor means and said second processor means recombine a fast Fourier transform of N complex points into a N real point fast Fourier transform in response to fourth control signals.

8. The apparatus of Claim 7, **characterized in that** said first processor means and said second processor means multiply (Section 3.6 Instruction BMUL) two sets of complex numbers in response to fifth control signals, said first processor means and said second processor means for square and sum (Section 3.7 Instruction BSQSM) complex input digital signals in response to sixth control signals, said first and said second processor a logical AND operation (Section 3.14 Instruction BAND) of corresponding members of two sets of input signals in response to seventh control signals.

9. The apparatus of one of the preceding claims, **characterized by** first bus means (Fig. 2.0: D1,D2,D5,D6) and second bus means (Fig. 2.0: D3,D4,D7,D8), said first bus means coupling input terminals of said apparatus to a one of input terminals, said input register means and output terminals of said output register means in response to first bus control signals, said second bus means coupling output terminals of said apparatus to said input register means input terminals when said first bus is not coupled thereto and coupling output terminals of said apparatus to said output register means output terminals when said first bus means is not coupled thereto.

10. The apparatus of one of the preceding claims, **characterized by** a first memory means coupled to said apparatus input terminals and second memory means coupled to said apparatus output terminals (Fig. 1.4: Memory units on either side of CE), wherein said apparatus processes digital signal groups from said first memory means and stores said resulting signal groups in said second memory means, said apparatus processing said resulting signal groups stored in said second memory means and storing second resulting signal groups in said first memory means.

11. The apparatus of claim 10, **characterized in that** said first and said second memory means each include a first and a second memory portion (Fig. 1.4: Memory units are shown in groups of 2), said first and said second memory portion permitting said first and second memory means to store and retrieve logic signal groups simultaneously.

12. The apparatus of one of the claims 3 to 11, **characterized in that** said apparatus is implemented using pipelining techniques (Fig. 2.12; 2.13; 2.14) having a cycle of Tm, said apparatus having a first mode of operation wherein two sets of digital signal groups are applied to said input register means during every Tm/4 period, said apparatus having a second mode of operation wherein four sets of digital signal groups are applied to input register means during every Tm/2 period.

13. The apparatus of Claim 12, **characterized in that** said Tm is less than or equal to 80 ns.

14. The apparatus of one of the claims 3 to 13, **characterized in that** said apparatus is fabricated on a single substrate using integrated circuit techniques.

15. The apparatus of Claim 14, **characterized in that** a latency associated with said apparatus is a 4Tm period.

16. Apparatus for processing digital signals, **characterized by:**

a) a plurality of adder units (Fig. 2.13: PS0-PS15) for combining a plurality of digital adder input digital signal groups according to an algorithm determined by first control signals applied thereto;

b) a plurality of multiplier units (Fig. 2.14: MUL0-MUL7) for multiplying first multiplier input digital signal groups by second multiplier input digital signal groups, wherein said first and said second digital signal groups represent complex numbers; and

c) a plurality of arithmetic logic units (Fig. 2.14; ALU0-ALU3), each arithmetic logic unit receiving digital signal groups from a coupled multiplier unit, said arithmetic logic unit forming said digital signal groups from said coupled multiplier unit into real and imaginary parts, wherein said apparatus performs a decimation-in-frequency butterfly algorithm when apparatus input signal groups are applied to said adder units and adder unit output signal groups are applied to said multiplier units, wherein said apparatus performs decimation-in-time algorithm when said apparatus signal groups are applied to said multiplier units and output signal groups from said arithmetic logic units are applied to said arithmetic units.

17. The apparatus of Claim 16, **characterized in that** said apparatus is fabricated in a pipelined implementation (Fig. 2.12, 2.13; 2.14) on a single substrate using integrated circuit techniques, each portion of said piplined implementation adapted to in a period Tm where Tm = 80 ns or less.

18. The apparatus of claim 16 or 17, **character-by** having a first mode of operation wherein two digital signal groups are applied to input terminals of said apparatus each Tm/4 ns and having a second mode of operation wherein four digital signal groups are applied to said apparatus input terminals each Tm/2 (Fig. 1.1).

19. The apparatus of Claim 18, **characterized in that** said apparatus can complete a radix-4 butterfly algorithm (Section 3.1 Instruction BFLY4), each Tm ns in response to first control signals and wherein said apparatus can complete two radix-2 butterfly algorithms (Section 3.2 Instruction BFLY2), each Tm ns in response to second control signals, said second multiplier digital signal groups being scaling factors.

20. The apparatus of one of the claims 16 to 19, **characterized by** first bus means (Fig. 2.0: D1, D2, D5, D6) for electrically coupling input terminals of said apparatus to input terminals of said adder units in response to a first bus signal, said first bus means coupling said apparatus input terminals to output terminals of said arithmetic logic units in response to a second bus signal; and including second bus means (Fig. 2.0: D3, D4, D7, D8), said second bus means coupling apparatus output terminals to said adder unit input terminals when said first bus means is not electrically coupled thereto.

21. The apparatus of Claim 18, 19 or 20, **characterized in that** said apparatus can complete algorithms recombining N complex point fast Fourier transform (Section 3.3 Instruction FFT2N) into two N real point fast Fourier transforms in Tm ns in response to third control signals, wherein said apparatus can complete algorithms recombining N complex point fast Fourier transforms (Section 3.4 Instruction FFTNN) into 2N real point fast Fourier transforms.

22. The apparatus of one of the claims 18 to 21, **characterized in that** said apparatus can execute block add algorithms (Section 3.8 Instruction BADD), block multiply algorithms (Section 3.6 Instruction BMUL) and block subtract algorithms (Section 3.9 Instruction BSUB) in response to preselected control signals.

23. The apparatus of one of the claims 18 to 22, **characterized in that** said apparatus can execute a block AND logic operation (Section 3.14 Instruction BAND), a block OR logic operation (Section 3.15 Instruciton BDK) and a block EXOR operation (Section 3.16 Instruction BXUR) in response to predetermined control signals.

24. The apparatus of one of the Claims 18 to 23, **characterized by** scale factor means, said scale factor means responsive to output signals from said arithmetic logic unit for providing a scaling factor, said scaling factor applied to signal groups from said arithmetic logic units prior to processing with a next consecutive butterfly algorithm.

25. The apparatus of one of the claims 16 to 24, **characterized by** first storage means coupled to input terminals of said apparatus and second storage means coupled to output terminals of said apparatus (Fig. 1.4 Memories on both sides of the CEs), wherein signal groups retrieved from said first storage means and stored in said second storage means can be processed by said apparatus and stored in said first storage means.

26. The apparatus of one of the claims 16 to 25, **characterized by** a second apparatus having a first and second storage means coupled thereto, said first apparatus along with coupled storage units being coupled to said apparatus and coupled storage means, said first appraatus performing a first operation on a block of signal groups, said second apparatus performing a next sequential operation on a block of signal groups resulting from said first operation (Fig. 1.4 Cascaded System).

27. A method of performing fast Fourier transforms on a block of digital signal groups, **characterized by** the steps of:

a) applying selected ones of said signal groups to adder units (Fig. 2.13: PS0-PS15);

b) combining said selected signal groups in said adder unit configuration determined by control signals;

c) applying output signal groups from said adder units to multiplier units (Fig. 2.14: MULO-MULT) and performing multiplicative operations determined by said control signals; and

d) combining output signal groups from said multiplier units in arithmetic logic units (Fig. 2.14: ALU0-ALU3) as determined by said control signals.

28. The method of Claim 27, **characterized by** the step of applying constant signal groups (Fig. 2.14: RX2(0)-RX2(7)) to said multiplier units to be multiplied by said adder unit output signal groups; and selecting said control signals to perform a radix-4 decimation-in-frequency algorithm in said adder units, said multiplier units and said arithmetic logic units.

29. The method of claim 27 or 28, **characterized by** performing it by means of a digital computing apparatus as described in one of the preceding claims.

TWO VERSIONS OF CE

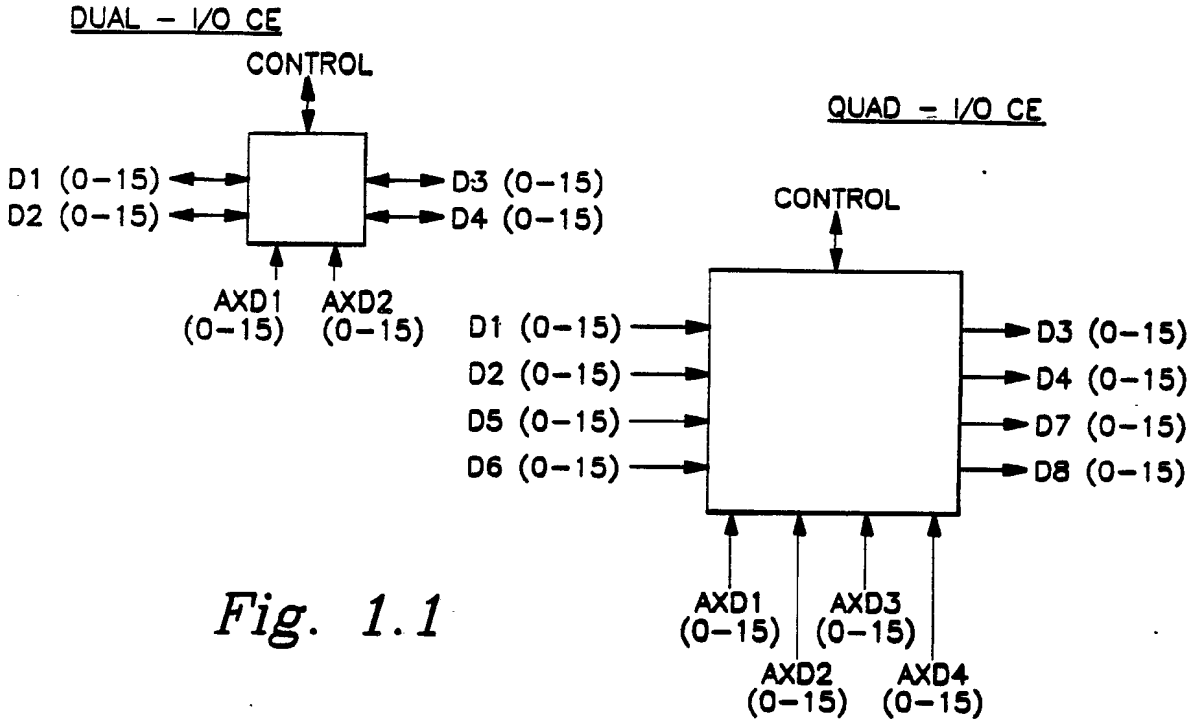


Fig. 1.1

MC – BLOCK DIAGRAM

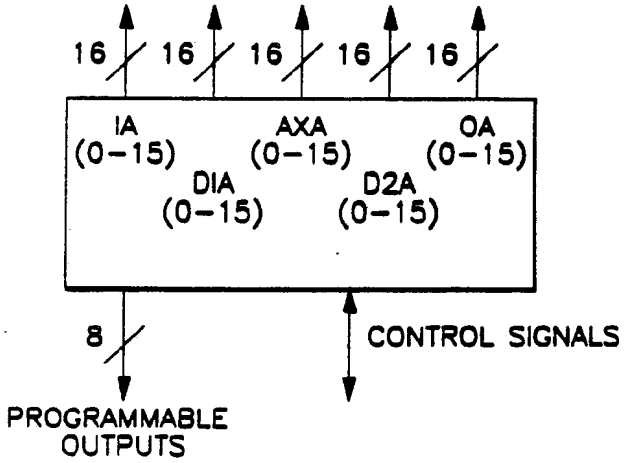


Fig. 1.2

DUEL – MEMORY (PING–PONG) SYSTEM

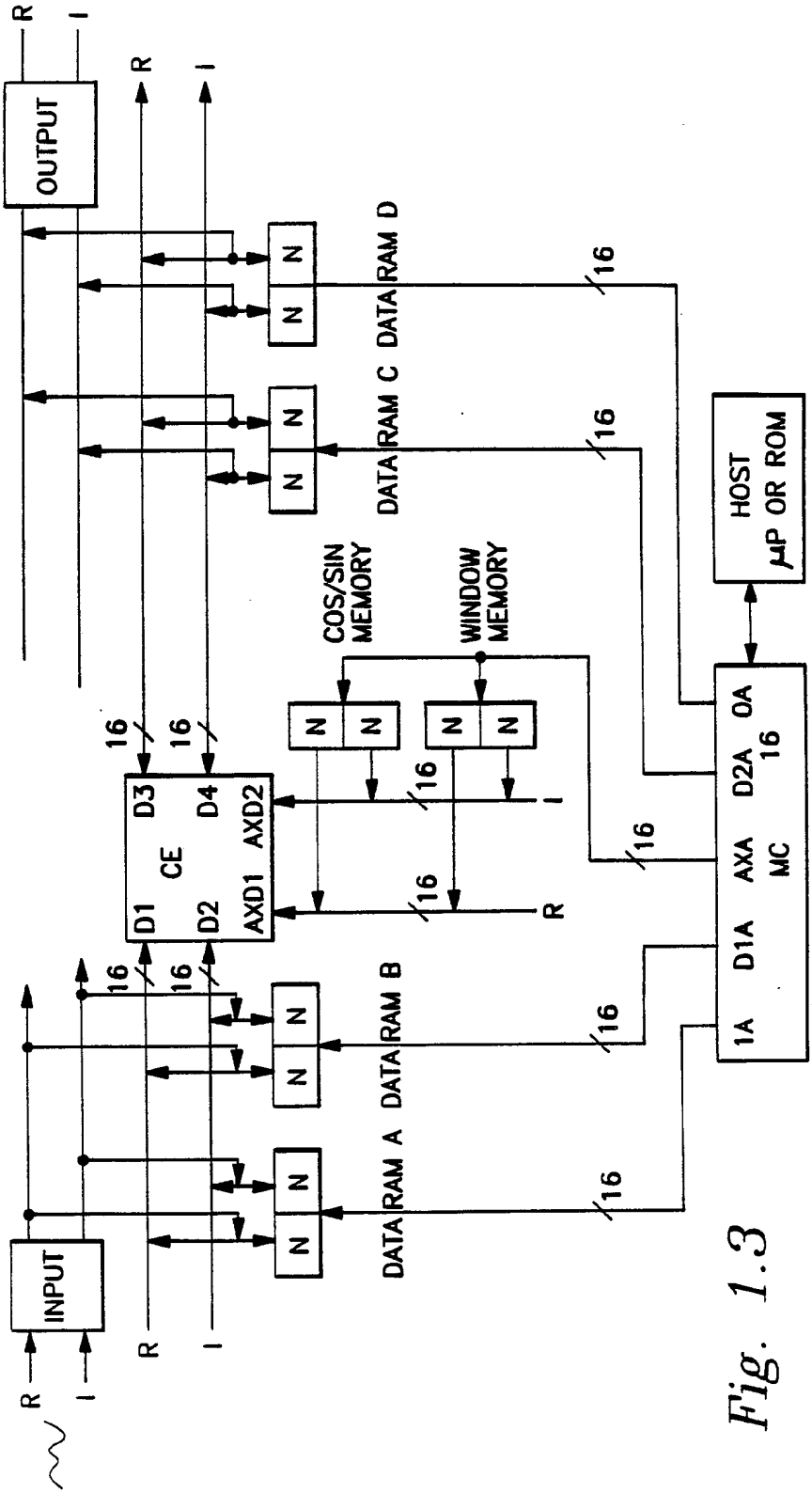


Fig. 1.3

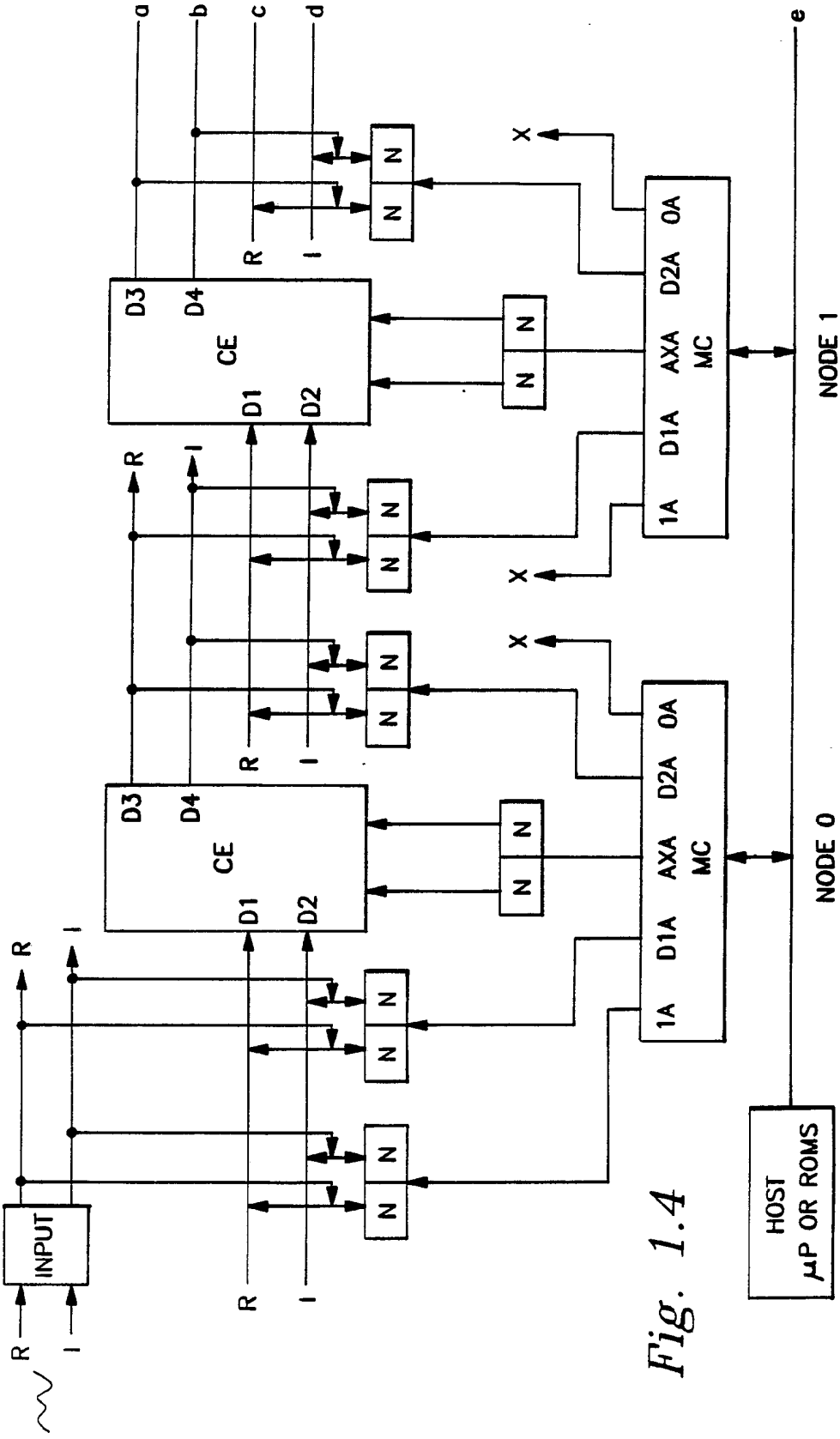

CASCADED – MEMORY FFT SYSTEM
INPUT
R
I
CE
D1
D2
D3
D4
N
1A
D1A
AXA
MC
D2A
0A
X
a
b
c
d
e
NODE 0
NODE 1
HOST
μP OR ROMS

Fig. 1.4

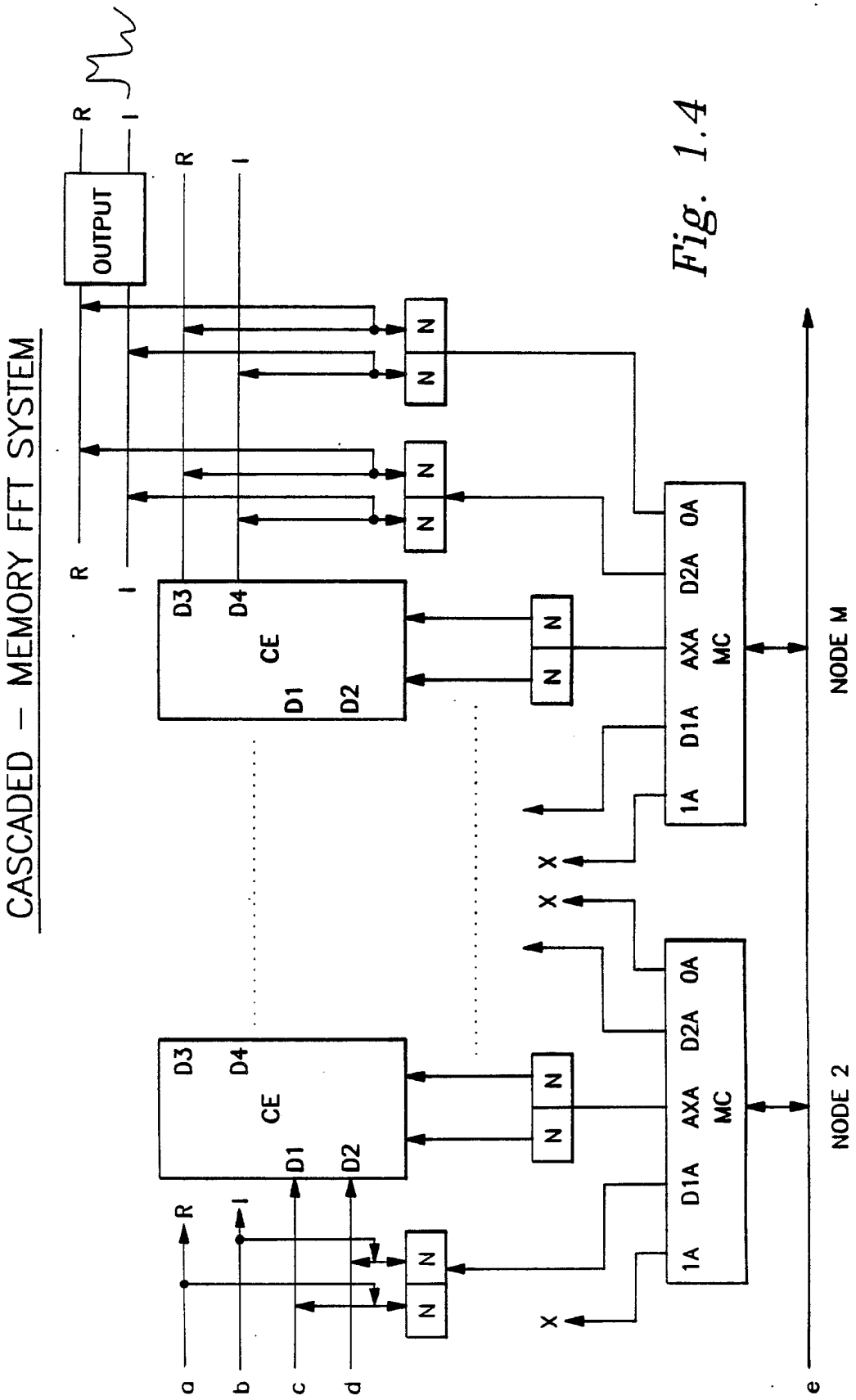

CASCADED – MEMORY FFT SYSTEM
OUTPUT
R
I
N
D3
D4
CE
D1
D2
0A
D2A
AXA
MC
D1A
1A
X
NODE M
NODE 2
a
b
c
d
e

Fig. 1.4

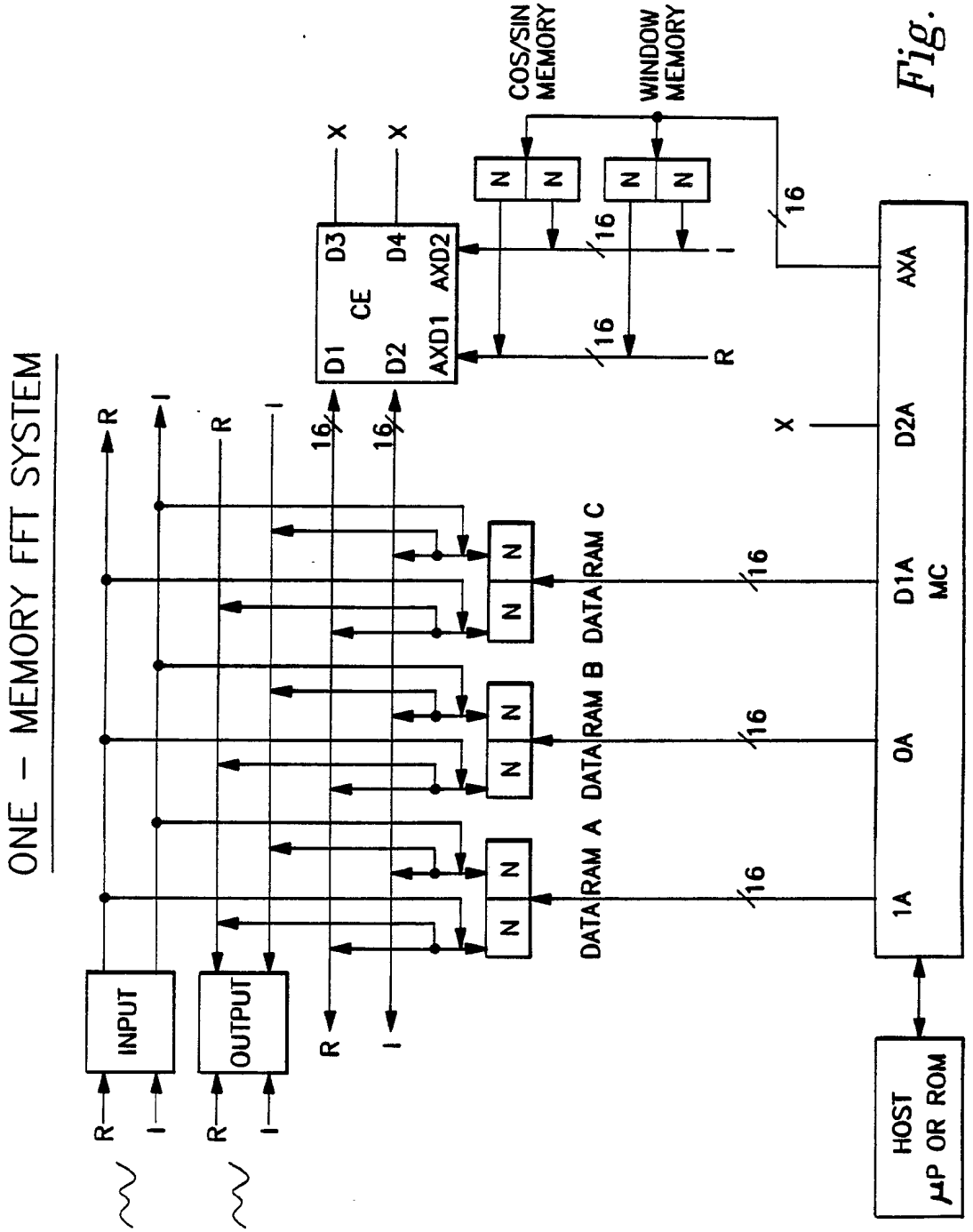

ONE – MEMORY FFT SYSTEM
INPUT
OUTPUT
R
I
16
CE
D1
D2
D3
D4
AXD1
AXD2
X
N
COS/SIN MEMORY
WINDOW MEMORY
DATA RAM A
DATA RAM B
DATA RAM C
1A
0A
D1A MC
D2A
AXA
HOST μP OR ROM

Fig. 1.5

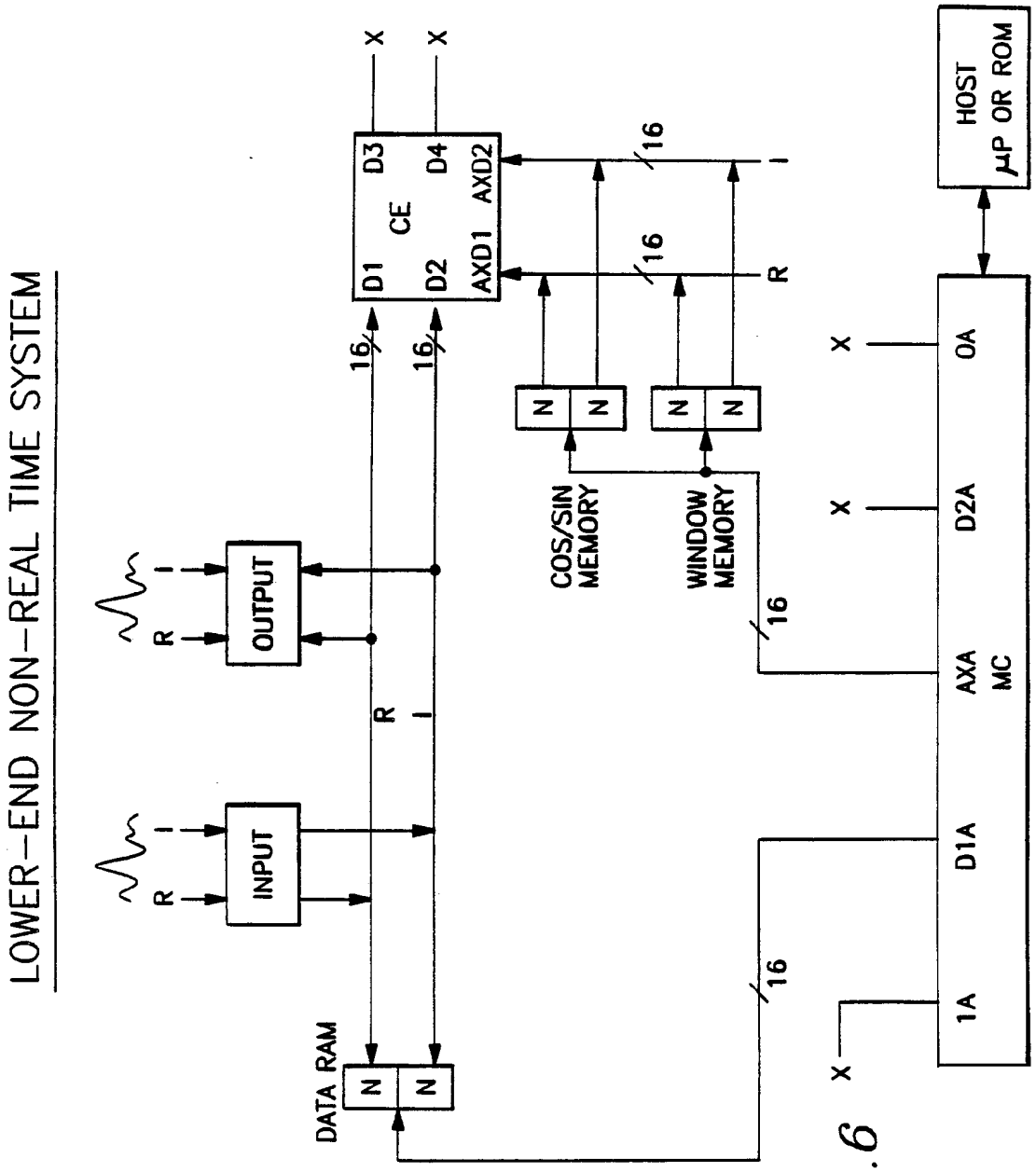

LOWER-END NON-REAL TIME SYSTEM
HOST
μP OR ROM
CE
D1
D2
D3
D4
AXD1
AXD2
OUTPUT
INPUT
R
I
N
16
X
COS/SIN
MEMORY
WINDOW
MEMORY
DATA RAM
0A
D2A
AXA
MC
D1A
1A

Fig. 1.6

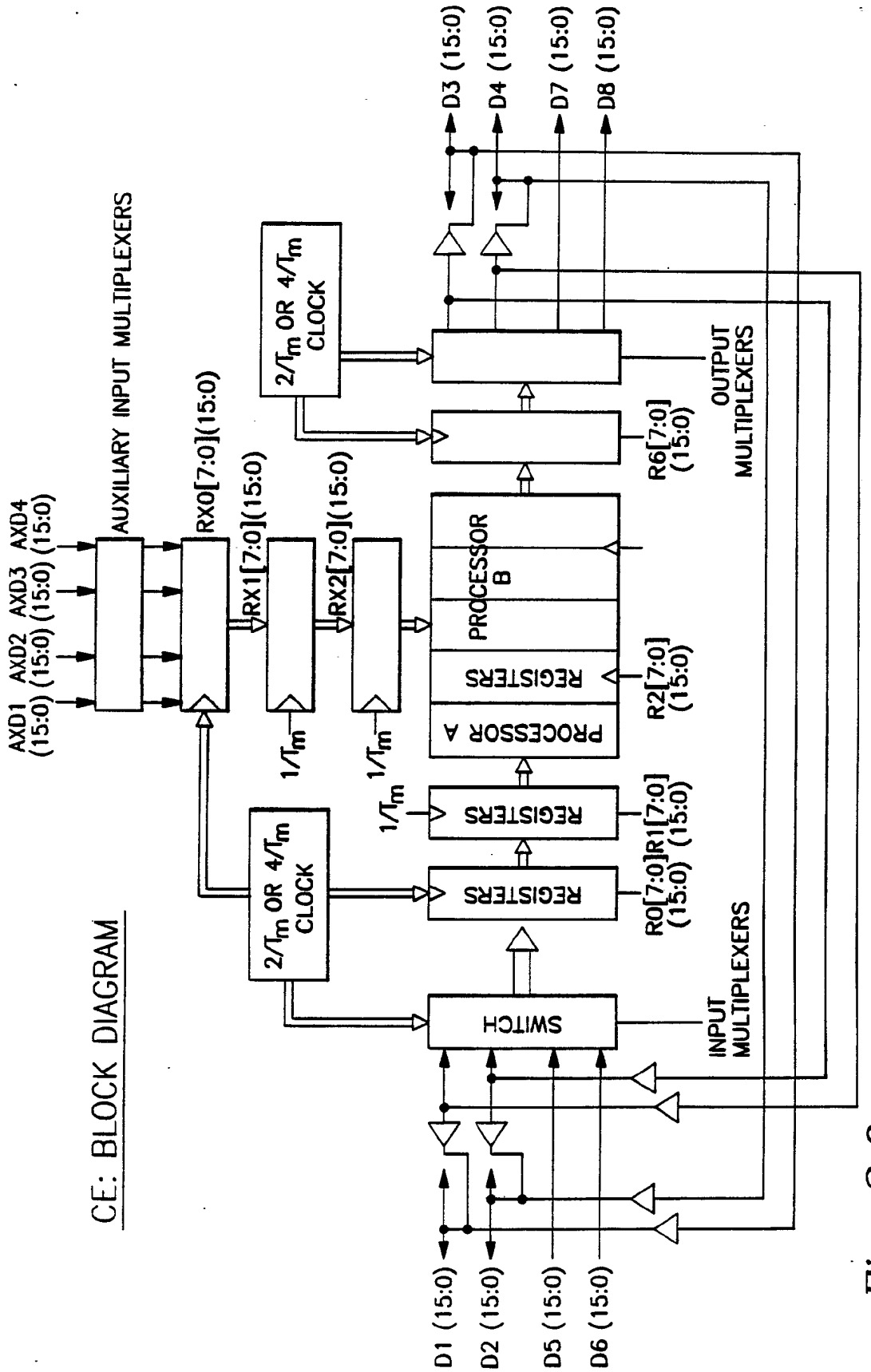

CE: BLOCK DIAGRAM
AXD1 (15:0)
AXD2 (15:0)
AXD3 (15:0)
AXD4 (15:0)
AUXILIARY INPUT MULTIPLEXERS
RX0[7:0](15:0)
RX1[7:0](15:0)
RX2[7:0](15:0)
1/Tm
1/Tm
1/Tm
2/Tm OR 4/Tm CLOCK
2/Tm OR 4/Tm CLOCK
REGISTERS
REGISTERS
REGISTERS
PROCESSOR A
PROCESSOR B
SWITCH
R0[7:0] (15:0)
R1[7:0] (15:0)
R2[7:0] (15:0)
R6[7:0] (15:0)
INPUT MULTIPLEXERS
OUTPUT MULTIPLEXERS
D1 (15:0)
D2 (15:0)
D5 (15:0)
D6 (15:0)
D3 (15:0)
D4 (15:0)
D7 (15:0)
D8 (15:0)

Fig. 2.0

CE: DUAL I/O MODE

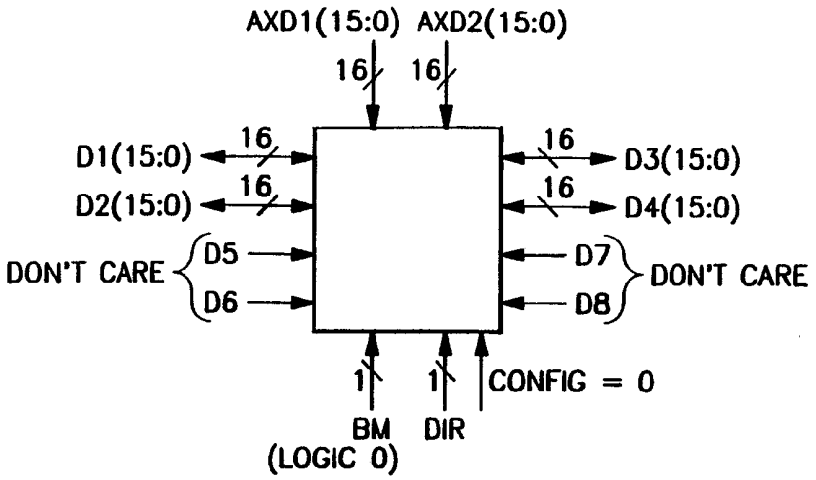


Fig. 2.1

CE: DUAL I/O MODE – DIRECTION CONTROL

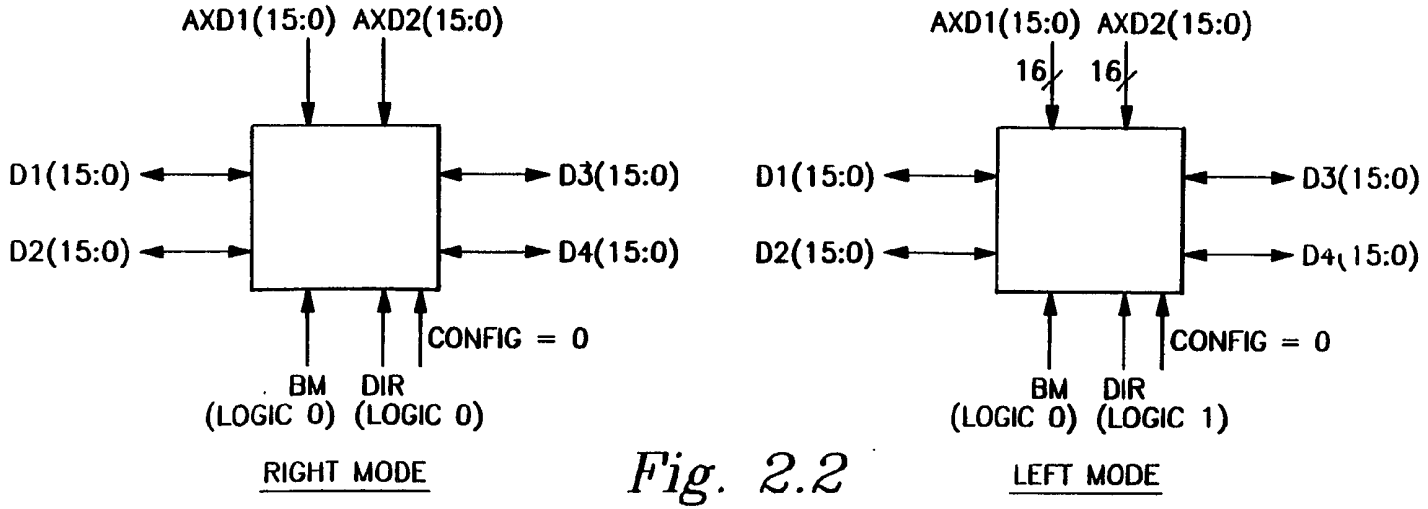


Fig. 2.2

DUAL I/O MODE – DIRECTION CONTROL AND CONFIG. PIN

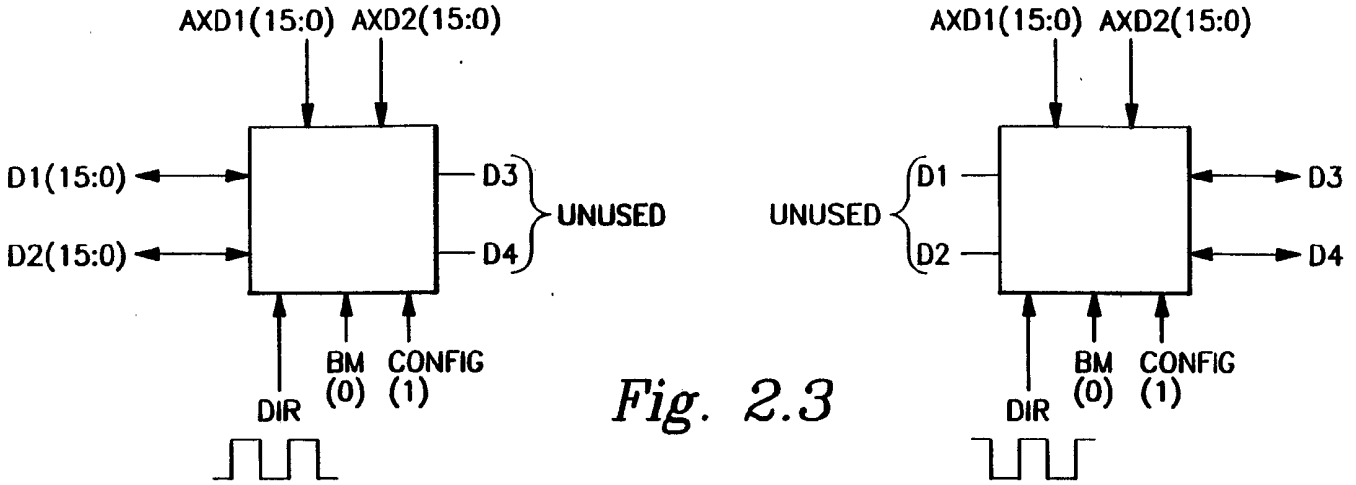


Fig. 2.3

CE: QUAD I/O MODE

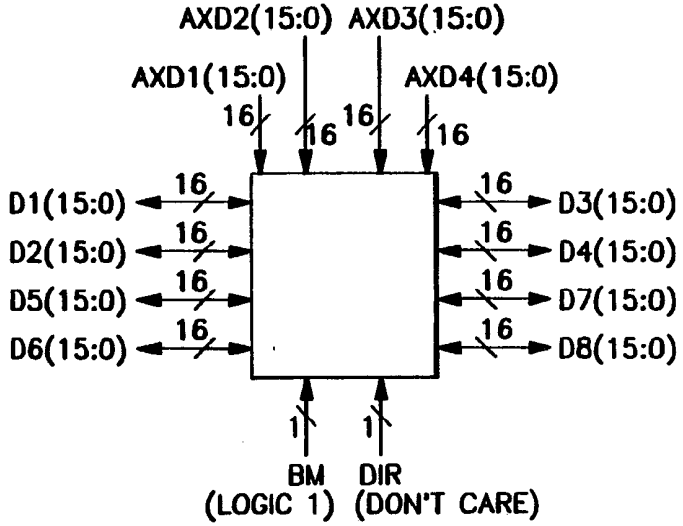


Fig. 2.4

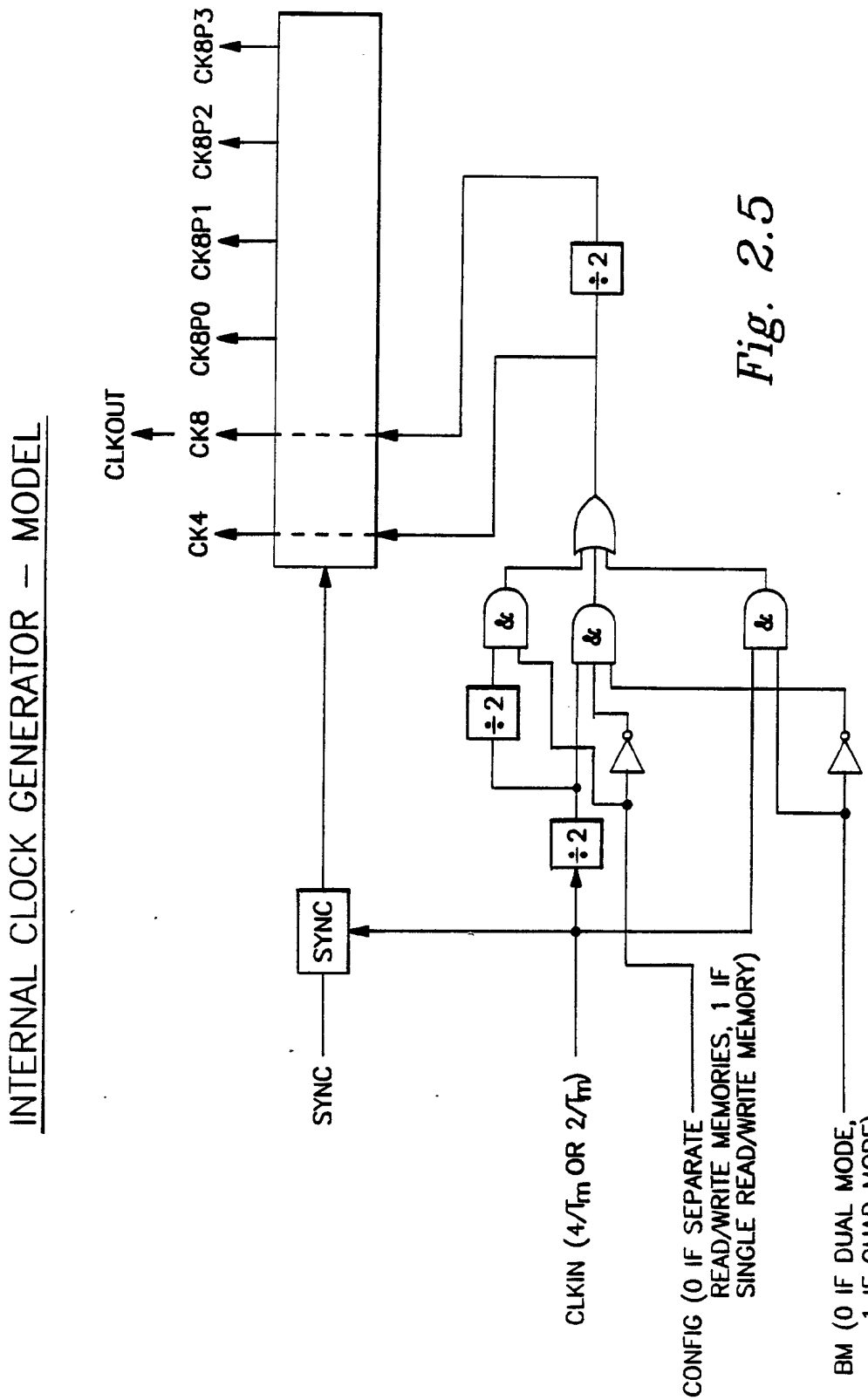

INTERNAL CLOCK GENERATOR – MODEL
CLKOUT
CK4
CK8
CK8P0
CK8P1
CK8P2
CK8P3
SYNC
÷2
÷2
÷2
&
&
&
SYNC
CLKIN (4/Tm OR 2/Tm)
CONFIG (0 IF SEPARATE READ/WRITE MEMORIES, 1 IF SINGLE READ/WRITE MEMORY)
BM (0 IF DUAL MODE, 1 IF QUAD MODE)

Fig. 2.5

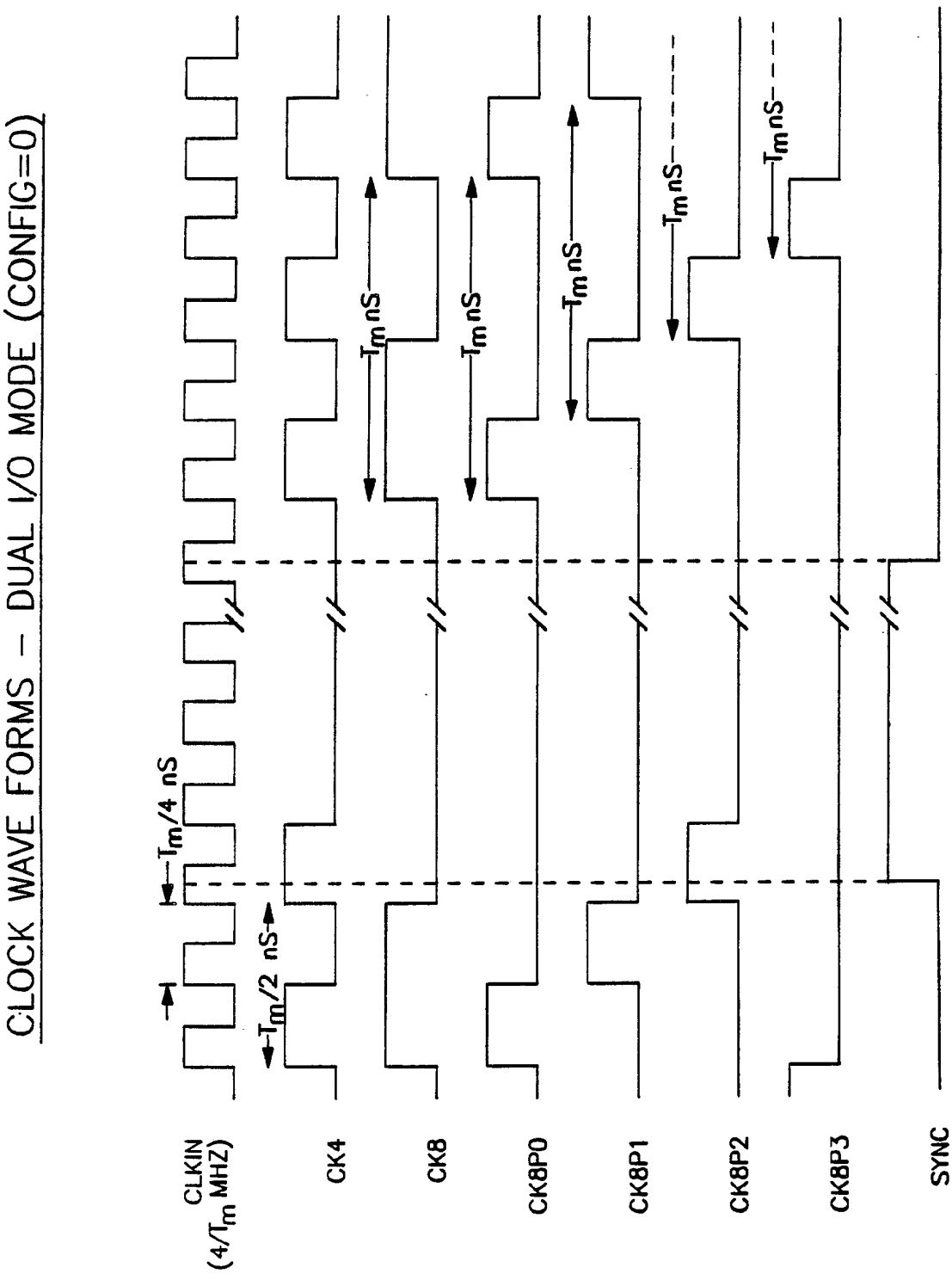

CLOCK WAVE FORMS – DUAL I/O MODE (CONFIG=0)
CLKIN
(4/Tm MHZ)
Tm/4 nS
Tm/2 nS
CK4
CK8
CK8P0
CK8P1
CK8P2
CK8P3
SYNC
Tm nS
Tm nS
Tm nS
Tm nS
Tm nS

Fig. 2.6

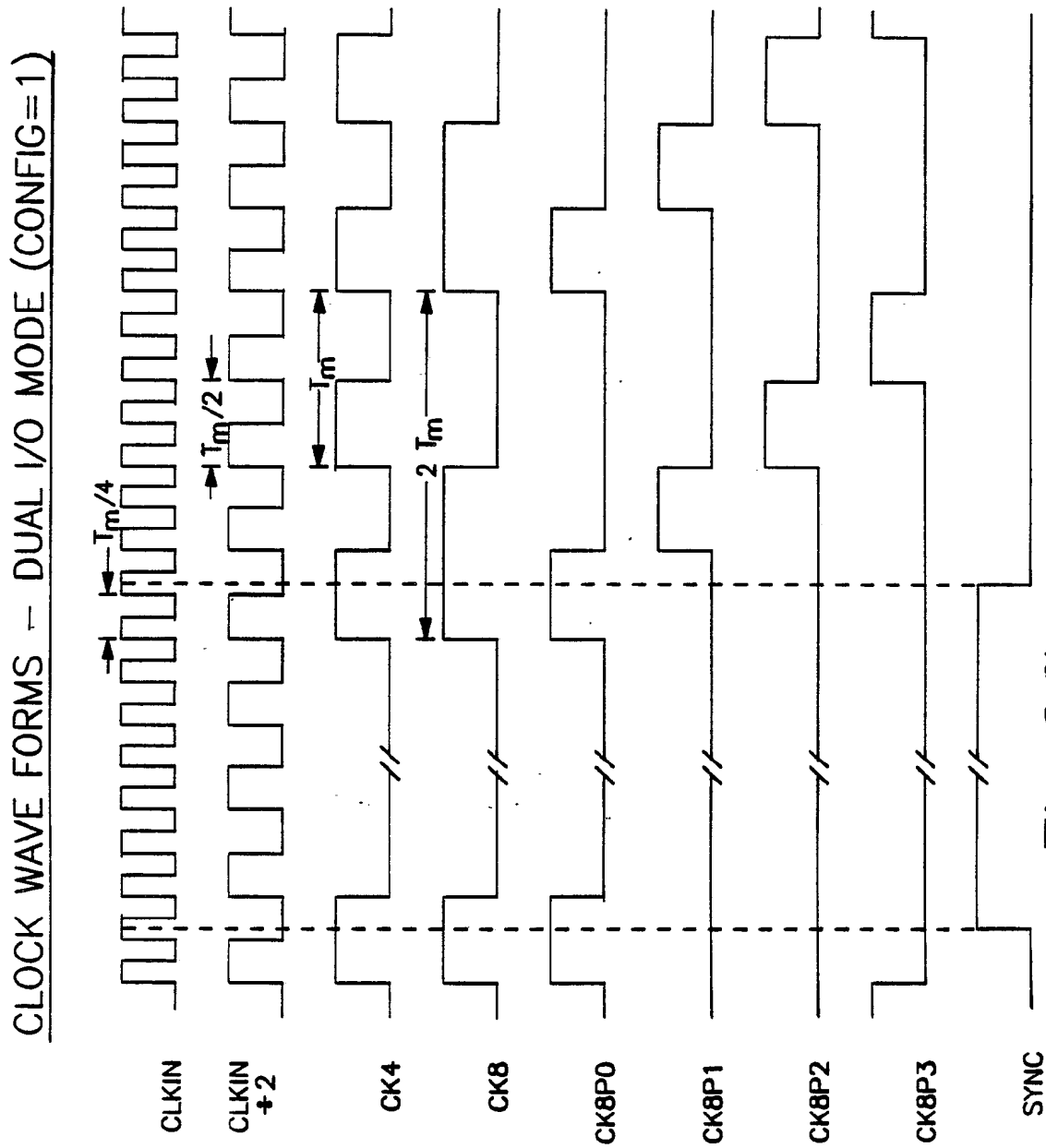

CLOCK WAVE FORMS – DUAL I/O MODE (CONFIG=1)
CLKIN
CLKIN +2
CK4
CK8
CK8P0
CK8P1
CK8P2
CK8P3
SYNC
Tm/4
Tm/2
Tm
2 Tm

Fig. 2.7

CLOCK WAVE FORMS – QUAD I/O MODE

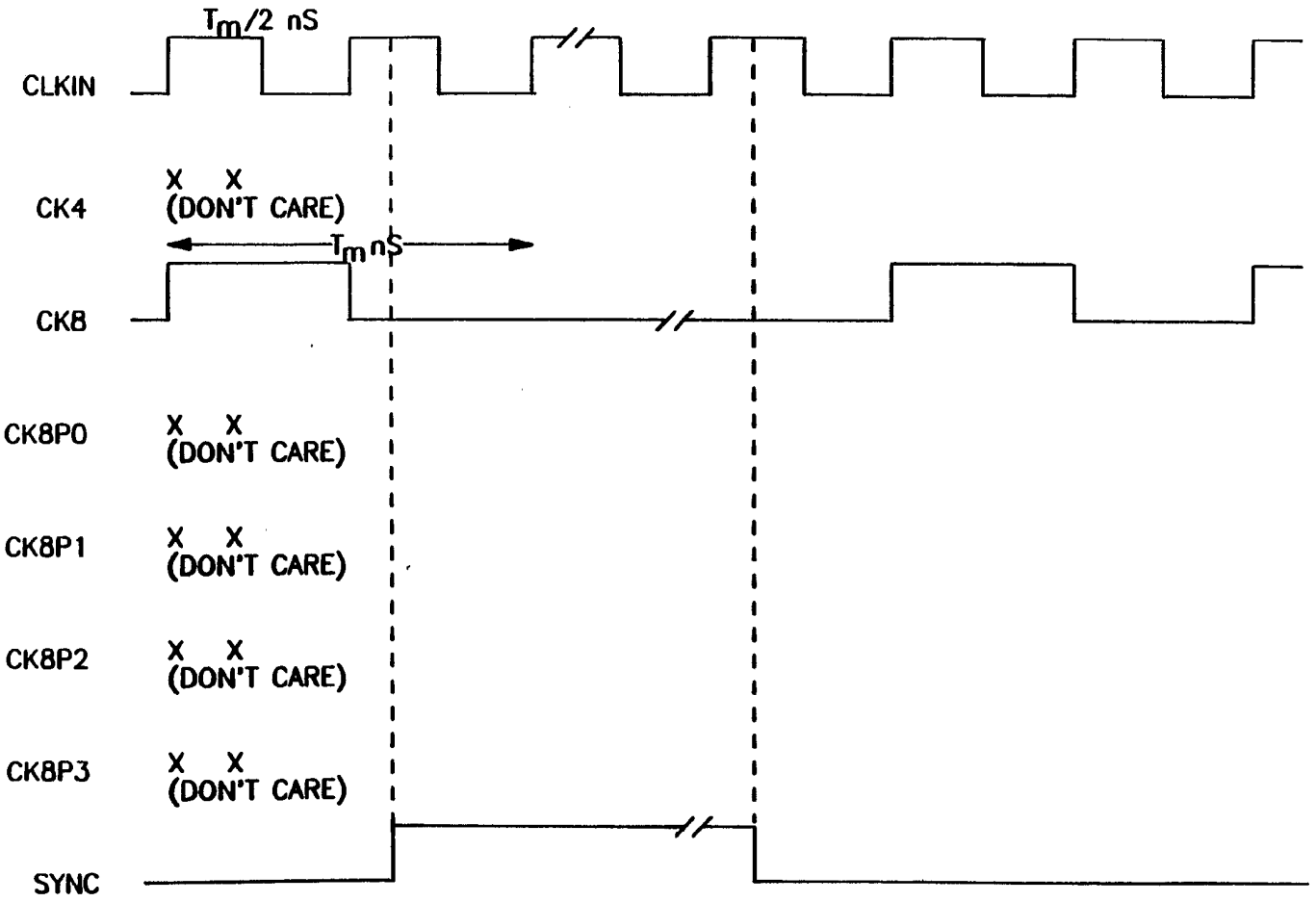


Fig. 2.8

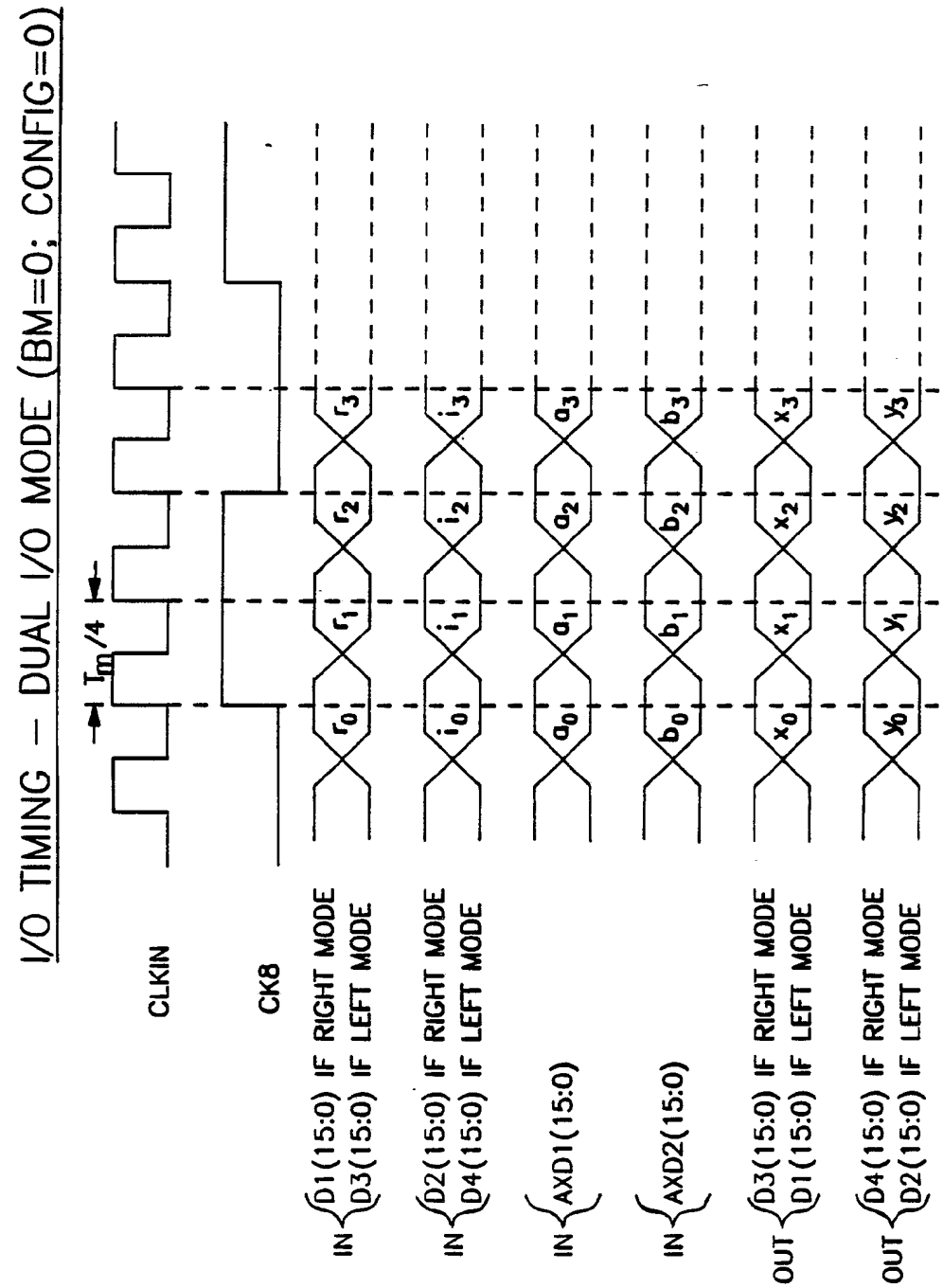
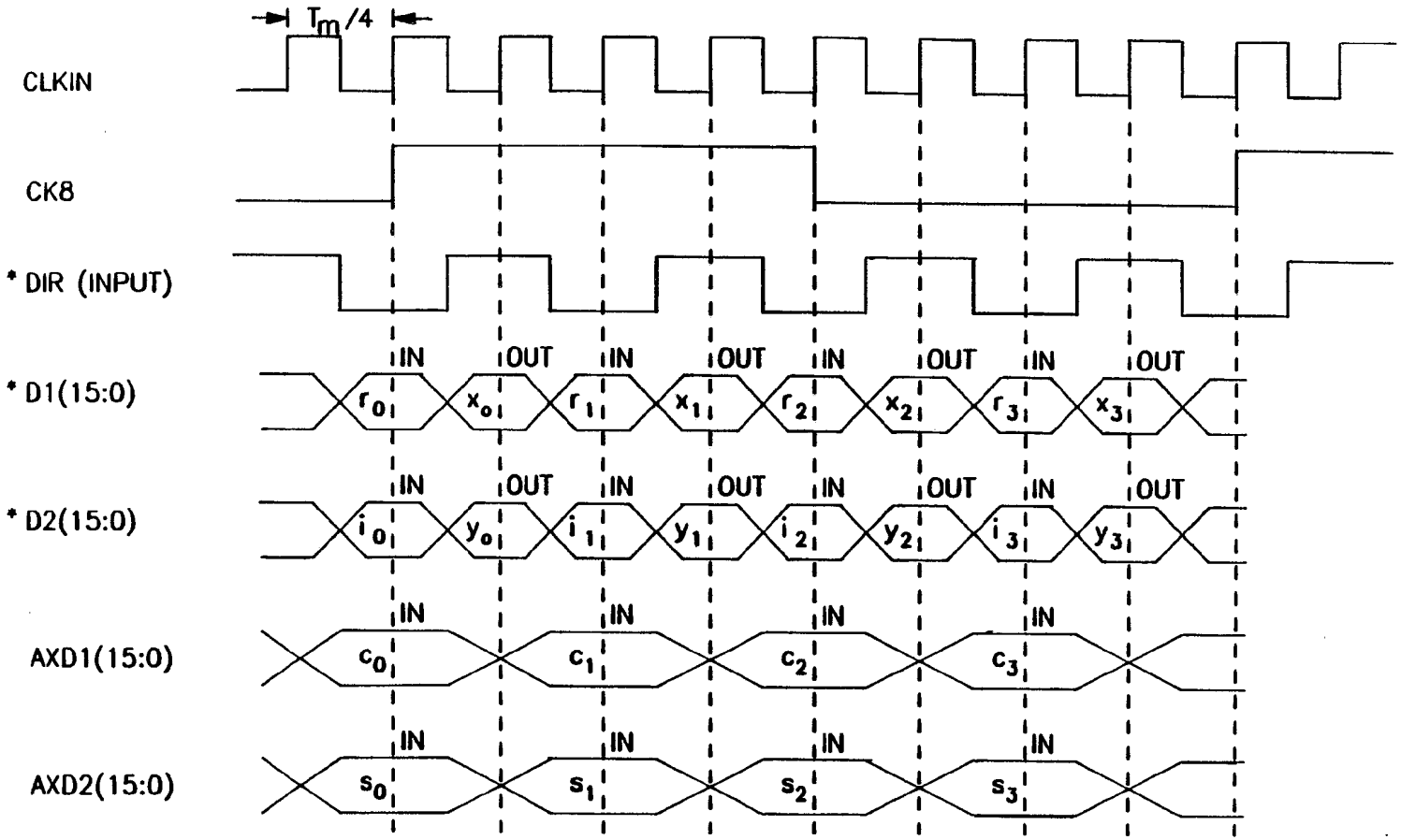

I/O TIMING – DUAL I/O MODE (BM=0; CONFIG=0)
CLKIN
CK8
$T_m/4$
IN D1(15:0) IF RIGHT MODE
D3(15:0) IF LEFT MODE
IN D2(15:0) IF RIGHT MODE
D4(15:0) IF LEFT MODE
IN AXD1(15:0)
IN AXD2(15:0)
OUT D3(15:0) IF RIGHT MODE
D1(15:0) IF LEFT MODE
OUT D4(15:0) IF RIGHT MODE
D2(15:0) IF LEFT MODE

Fig. 2.9

I/O TIMING – DUAL I/O MODE (BM=0, CONFIG=1)



*NOTE: BUSES D3 AND D4 COULD BE USED AS I/O BUSES (INSTEAD OF D1 AND D2) BY REVERSING THE POLARITY OF DIR SIGNAL SHOWN HERE

Fig. 2.10

I/O TIMING – QUAD I/O MODE (RM=1)

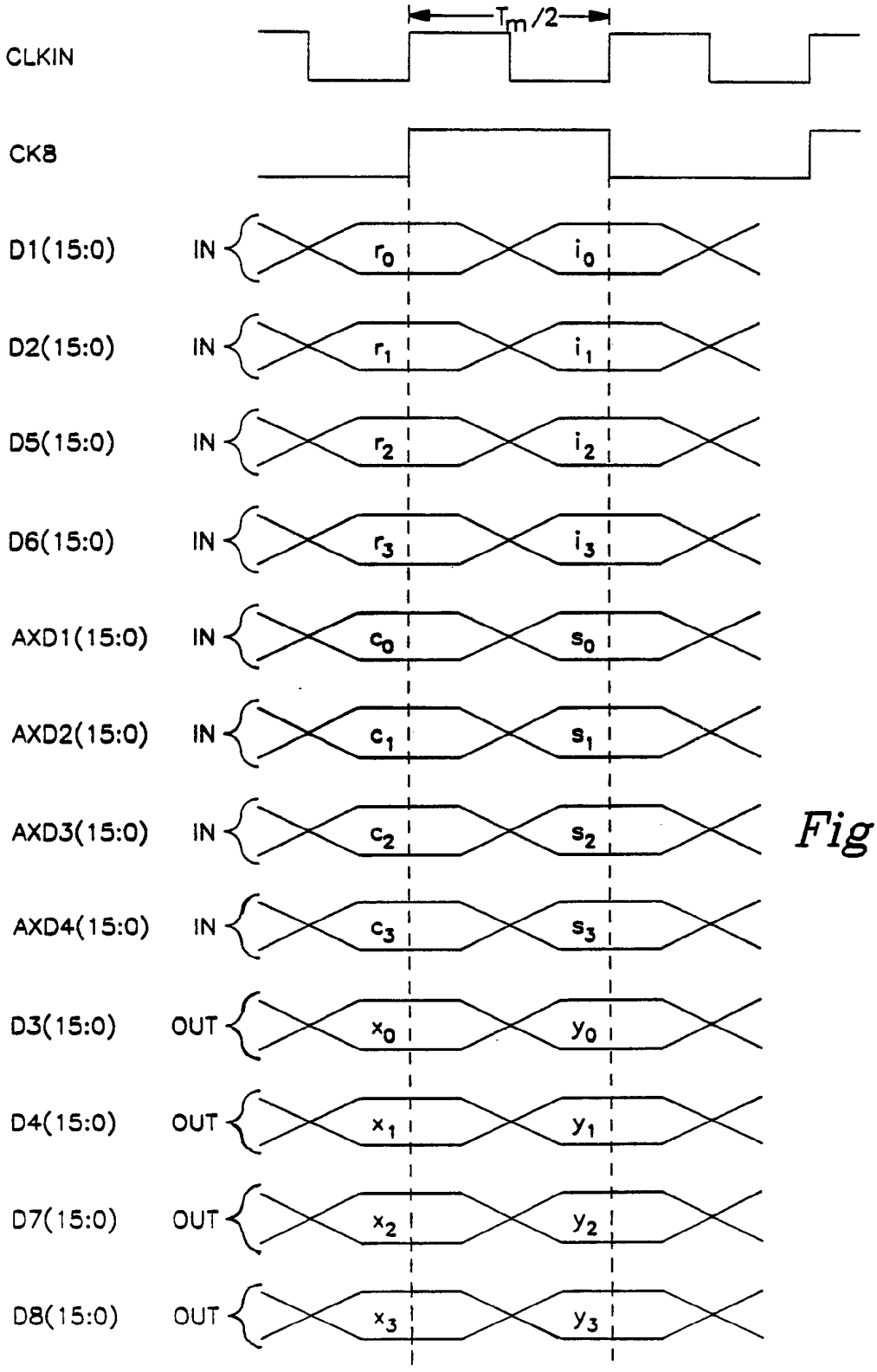


Fig. 2.11

Fig. 2.12a

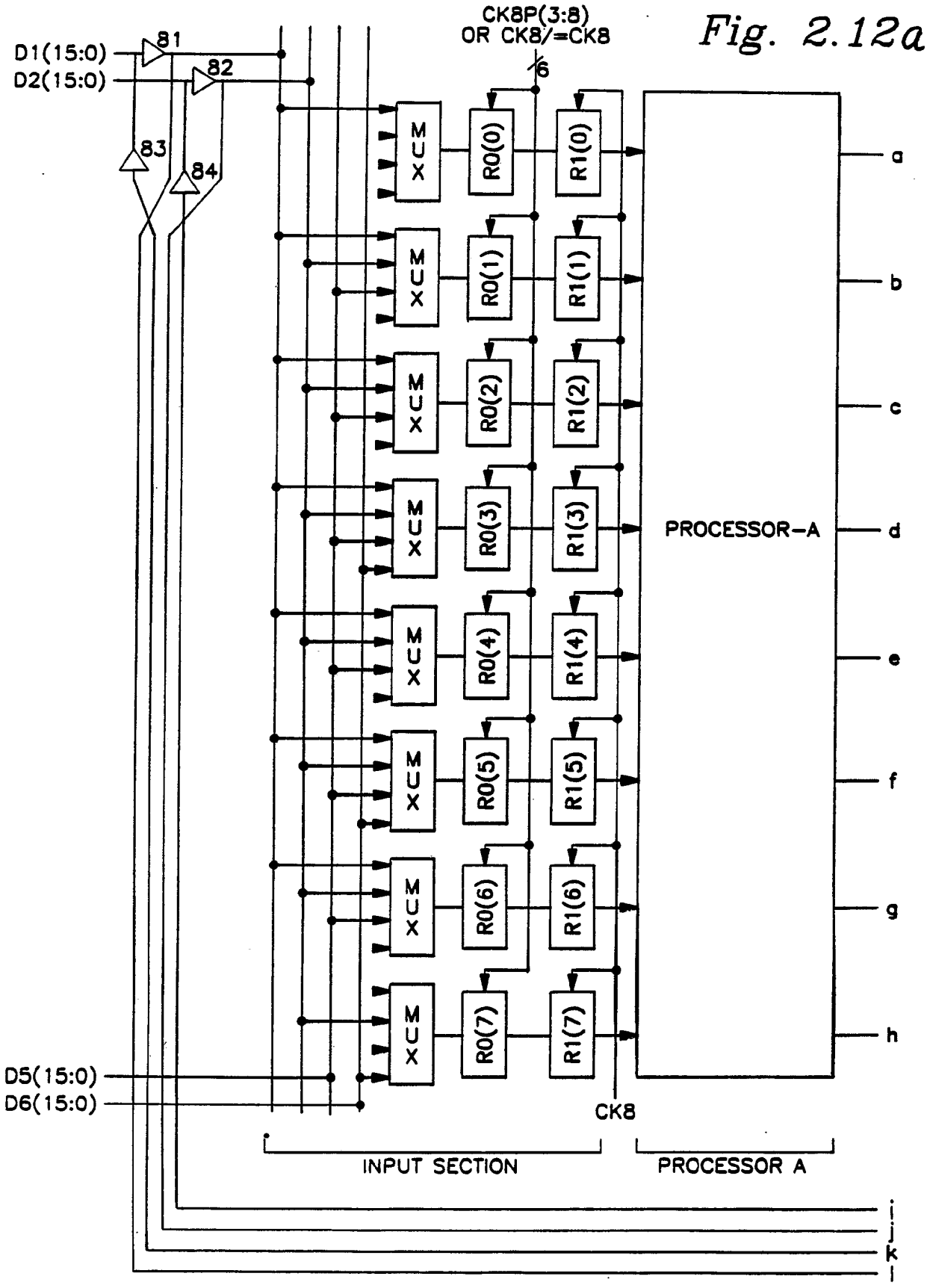

CK8P(3:8)
OR CK8/=CK8
6
D1(15:0)
D2(15:0)
81
82
83
84
MUX
R0(0)
R0(1)
R0(2)
R0(3)
R0(4)
R0(5)
R0(6)
R0(7)
R1(0)
R1(1)
R1(2)
R1(3)
R1(4)
R1(5)
R1(6)
R1(7)
PROCESSOR-A
a
b
c
d
e
f
g
h
D5(15:0)
D6(15:0)
CK8
INPUT SECTION
PROCESSOR A
i
j
k
l

Fig. 2.12b

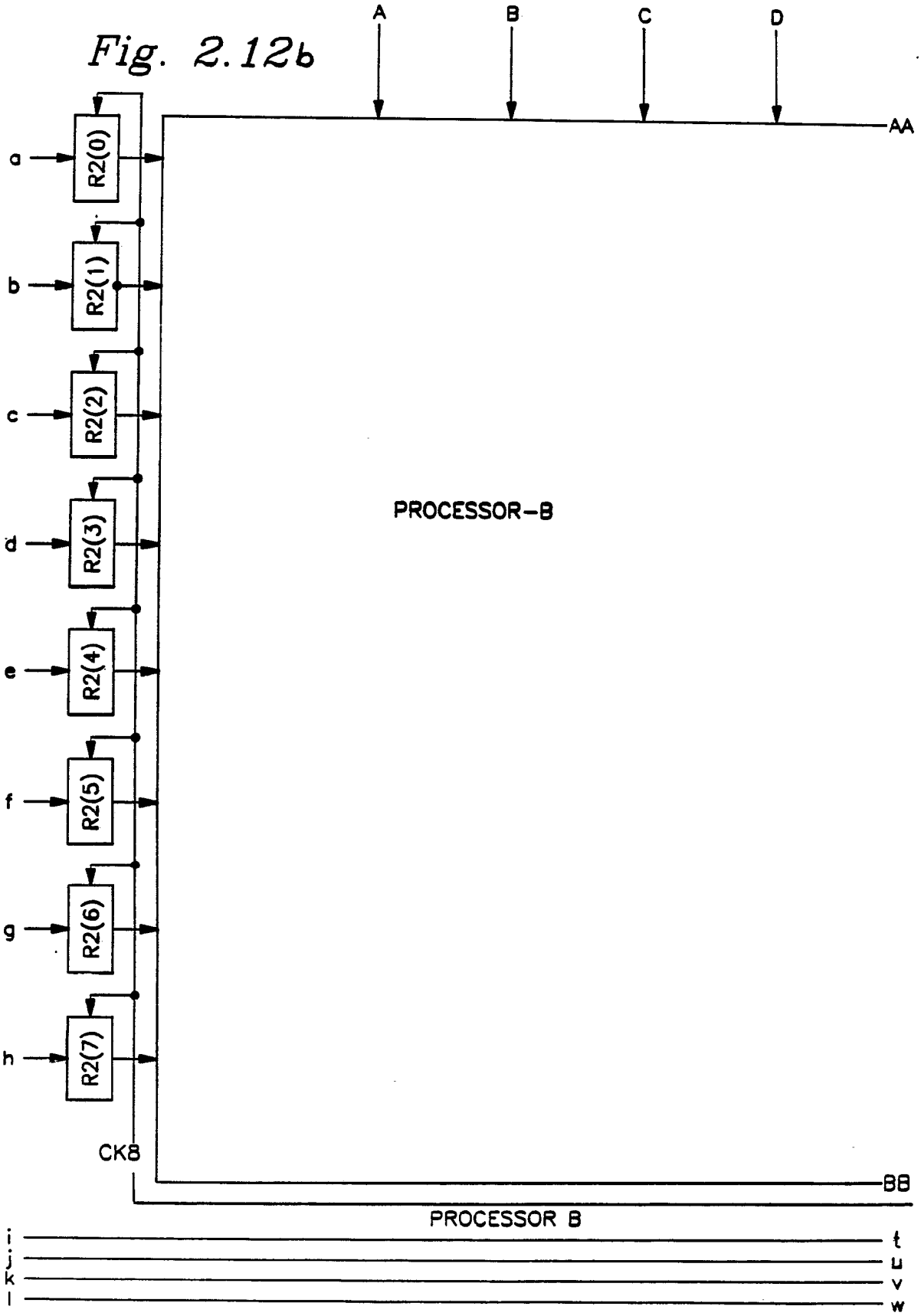
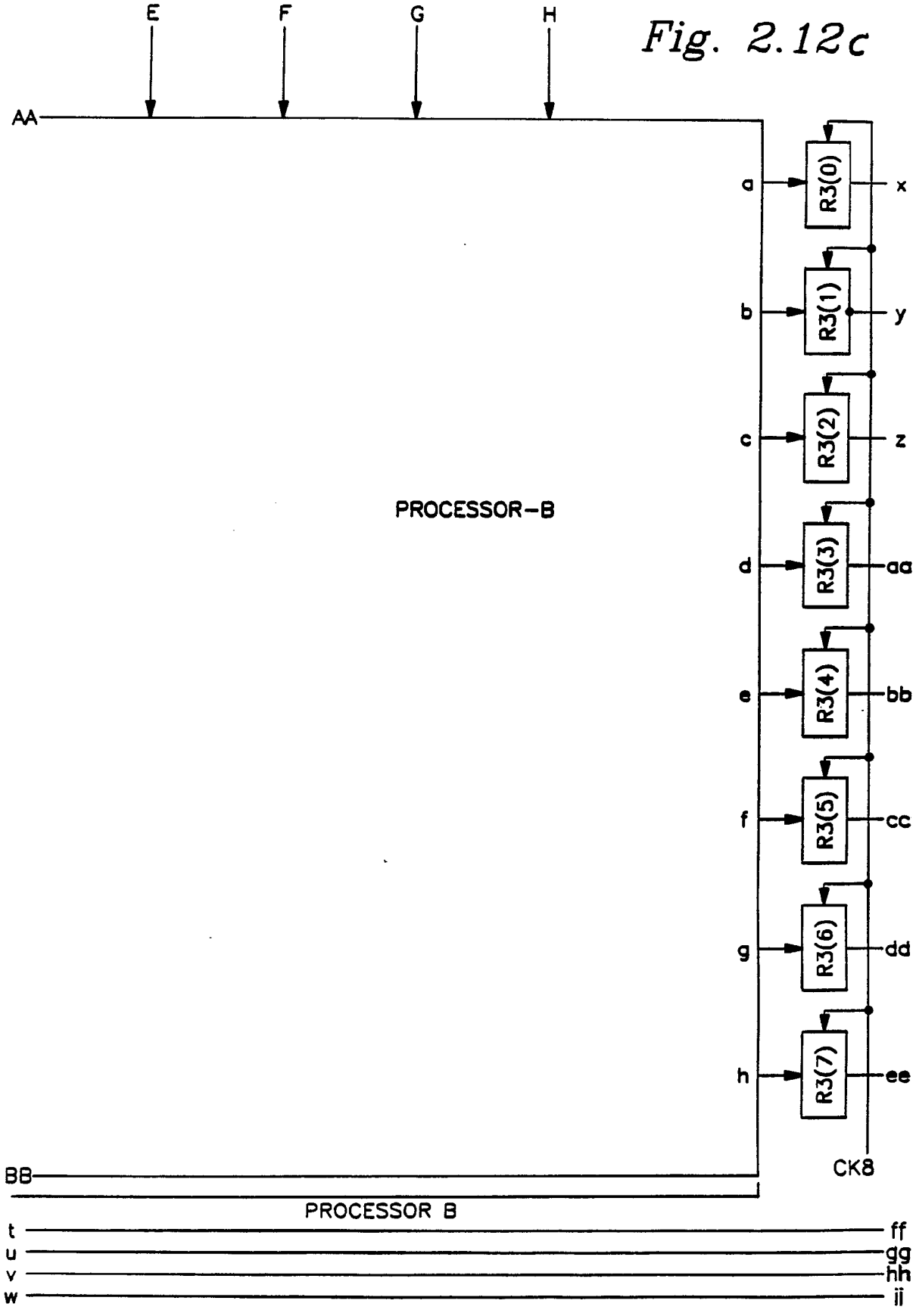

A
B
C
D
AA
a
b
c
d
e
f
g
h
R2(0)
R2(1)
R2(2)
R2(3)
R2(4)
R2(5)
R2(6)
R2(7)
PROCESSOR-B
CK8
BB
PROCESSOR B
i
j
k
l
t
u
v
w

Fig. 2.12c


E
F
G
H
AA
a
b
c
d
e
f
g
h
R3(0)
R3(1)
R3(2)
R3(3)
R3(4)
R3(5)
R3(6)
R3(7)
x
y
z
aa
bb
cc
dd
ee
PROCESSOR-B
BB
CK8
PROCESSOR B
t
u
v
w
ff
gg
hh
ii

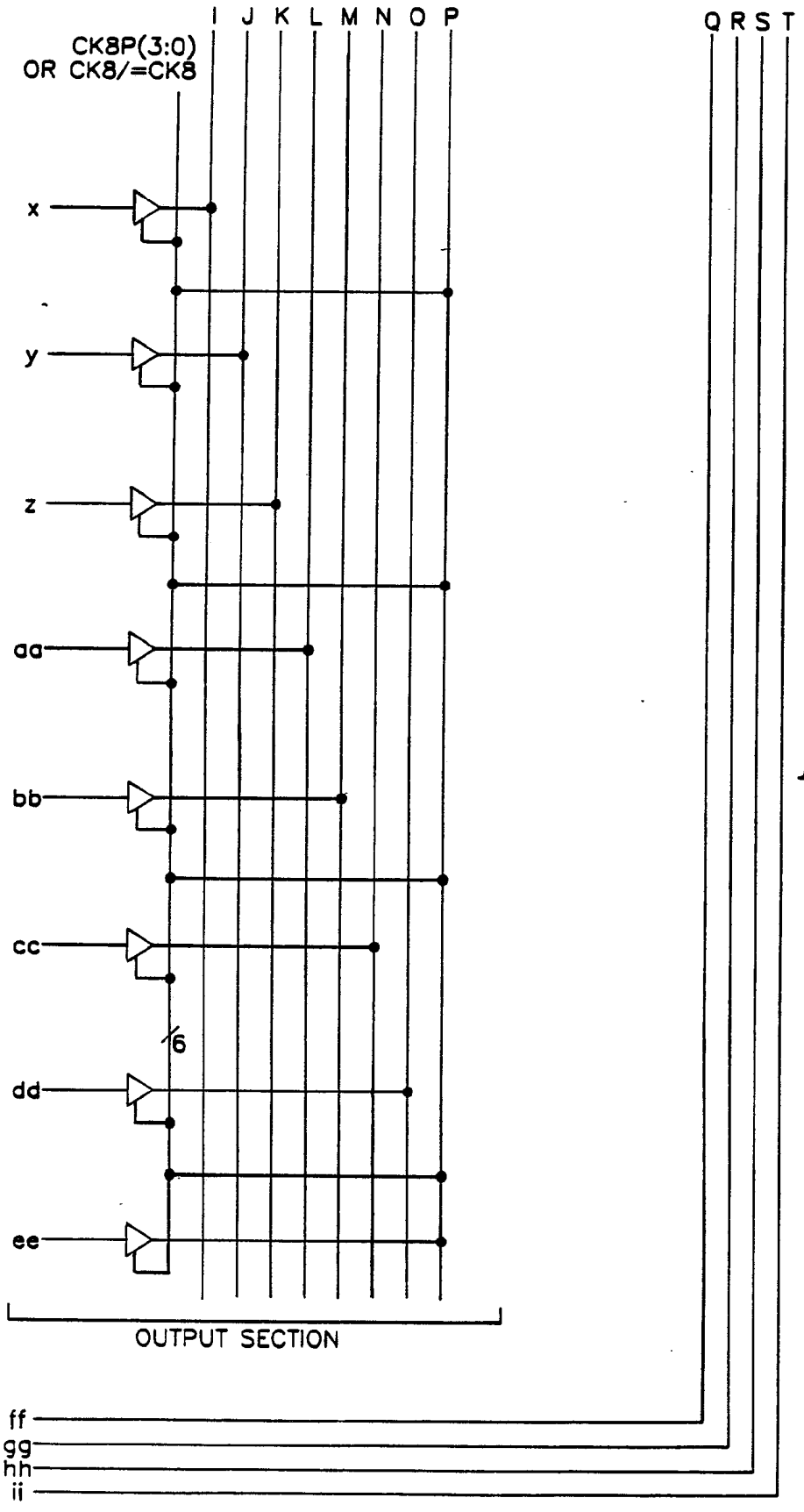

I J K L M N O P
Q R S T
CK8P(3:0)
OR CK8/=CK8
x
y
z
aa
bb
cc
/6
dd
ee
OUTPUT SECTION
ff
gg
hh
ii

Fig. 2.12d

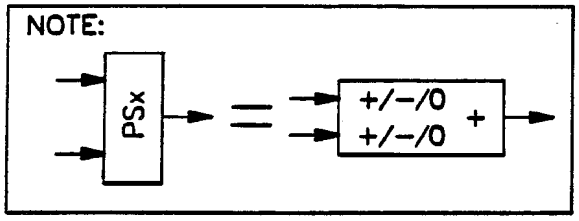

NOTE:
PSx
+/−/0
+/−/0
+

Fig. 2.12e

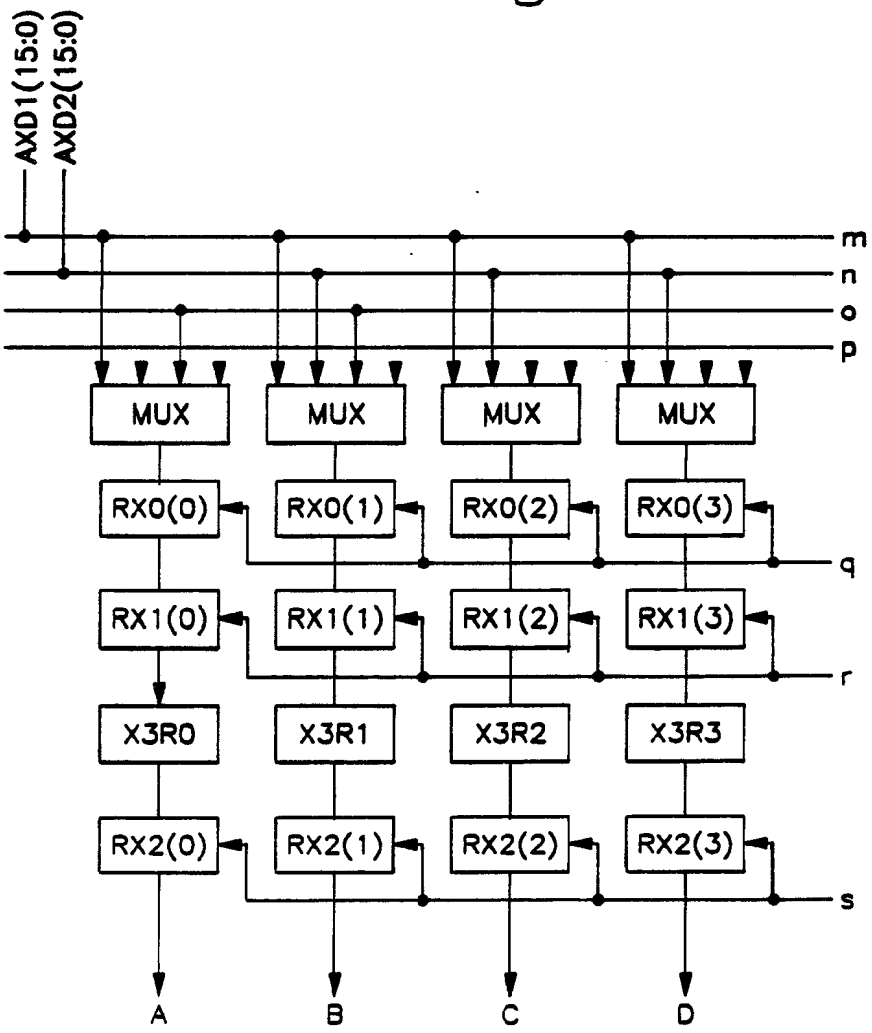

AXD1(15:0)
AXD2(15:0)
m
n
o
p
MUX
MUX
MUX
MUX
RX0(0)
RX0(1)
RX0(2)
RX0(3)
q
RX1(0)
RX1(1)
RX1(2)
RX1(3)
r
X3R0
X3R1
X3R2
X3R3
RX2(0)
RX2(1)
RX2(2)
RX2(3)
s
A
B
C
D

Fig. 2.12f

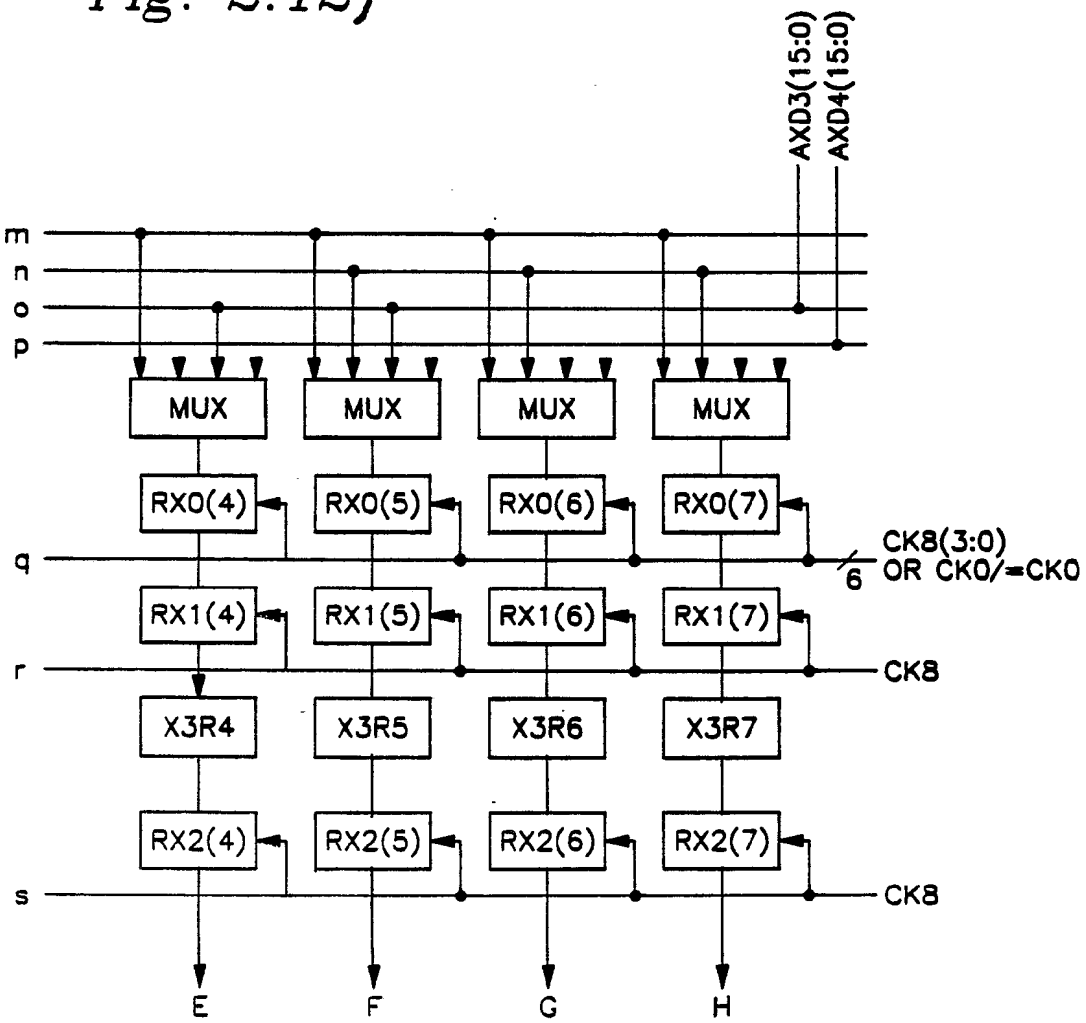
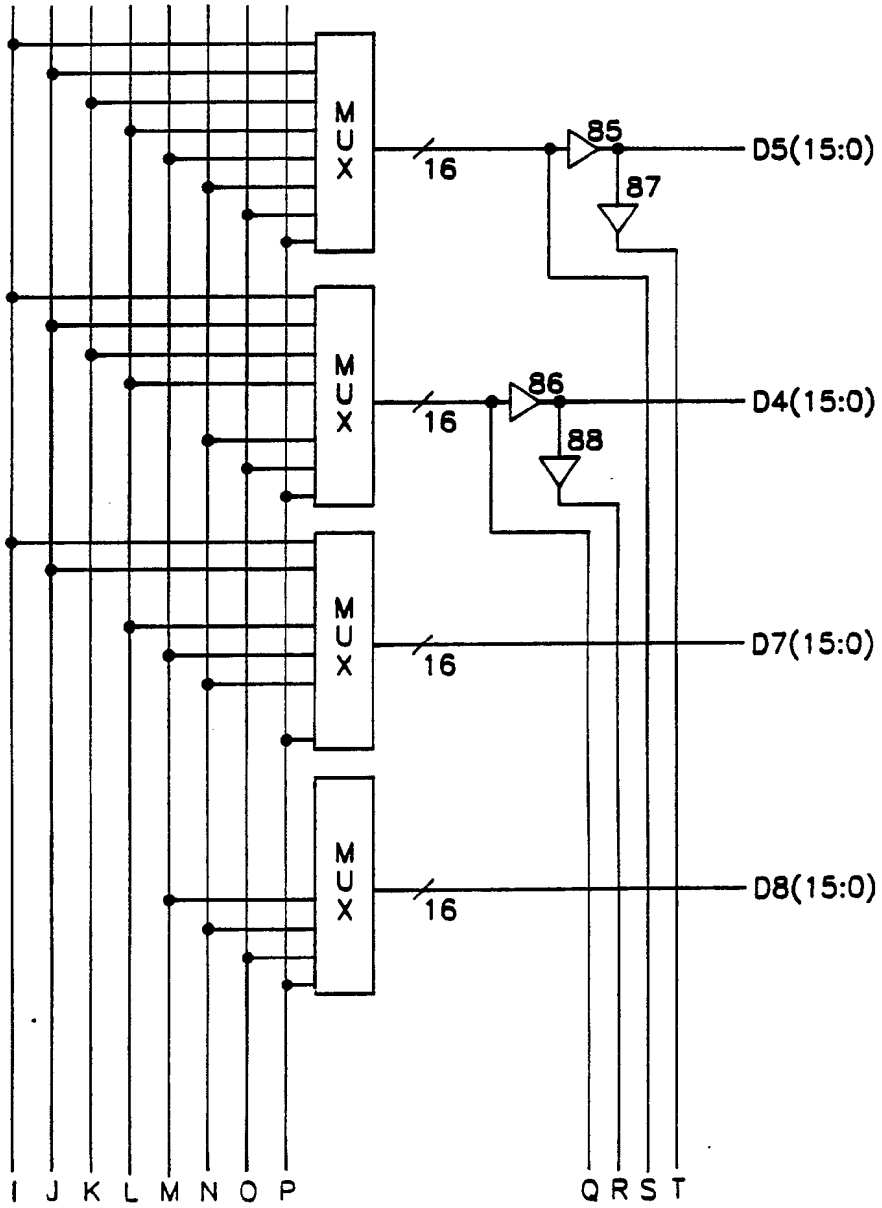

AXD3(15:0)
AXD4(15:0)
m
n
o
p
MUX
MUX
MUX
MUX
RX0(4)
RX0(5)
RX0(6)
RX0(7)
q
CK8(3:0)
OR CK0/=CK0
6
RX1(4)
RX1(5)
RX1(6)
RX1(7)
r
CK8
X3R4
X3R5
X3R6
X3R7
RX2(4)
RX2(5)
RX2(6)
RX2(7)
s
CK8
E
F
G
H
AUXILIARY
INPUT SECTION

Fig. 2.12g


MUX
MUX
MUX
MUX
16
16
16
16
85
86
87
88
D5(15:0)
D4(15:0)
D7(15:0)
D8(15:0)
I J K L M N O P
Q R S T

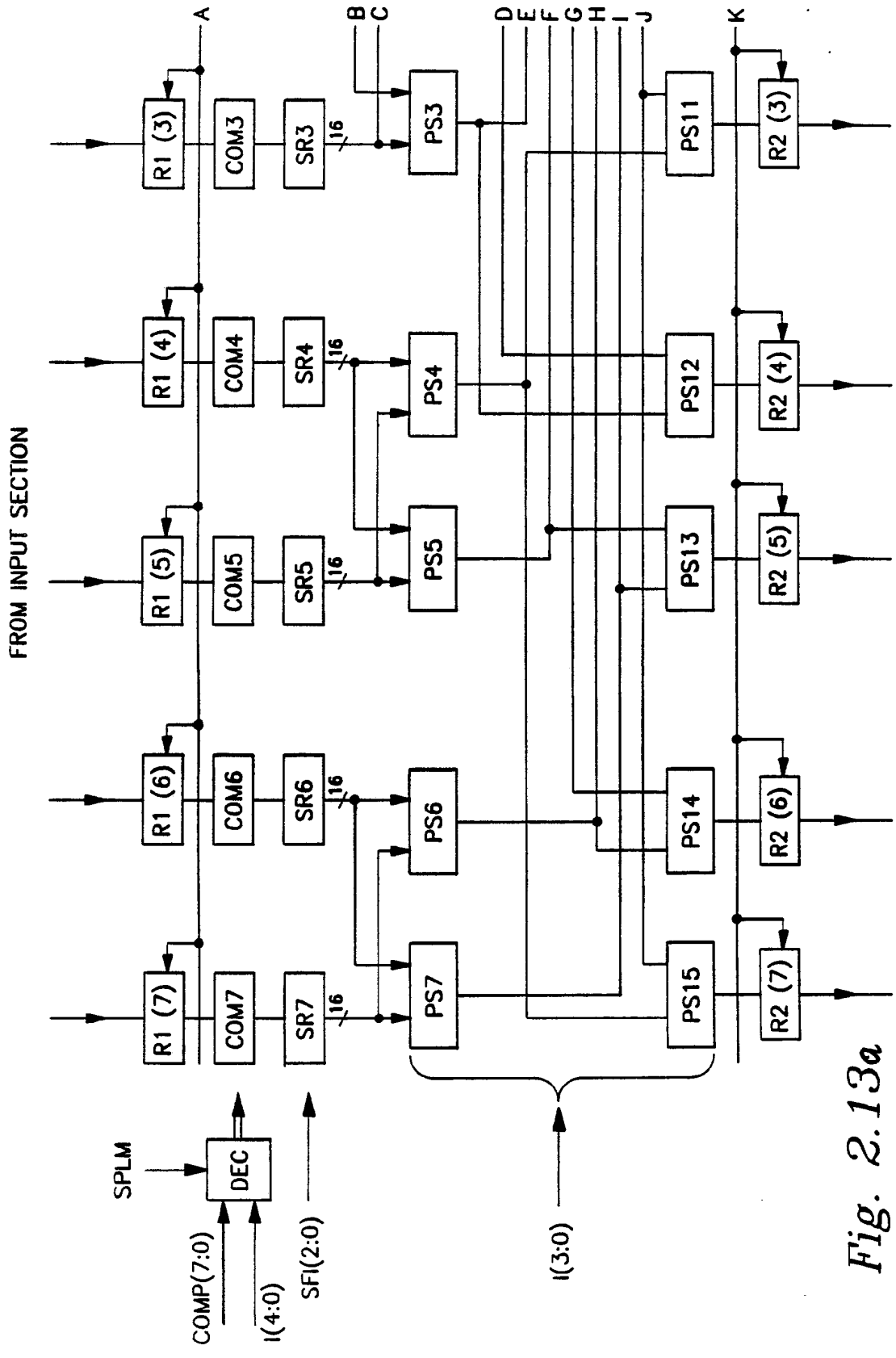

FROM INPUT SECTION
TO PROCESSOR-B
A
B
C
D
E
F
G
H
I
J
K
R1 (3)
R1 (4)
R1 (5)
R1 (6)
R1 (7)
COM3
COM4
COM5
COM6
COM7
SR3
SR4
SR5
SR6
SR7
16
PS3
PS4
PS5
PS6
PS7
PS11
PS12
PS13
PS14
PS15
R2 (3)
R2 (4)
R2 (5)
R2 (6)
R2 (7)
SPLM
DEC
COMP(7:0)
I(4:0)
SFI(2:0)
I(3:0)

Fig. 2.13a

PROCESSOR – A ARCHITECTURE

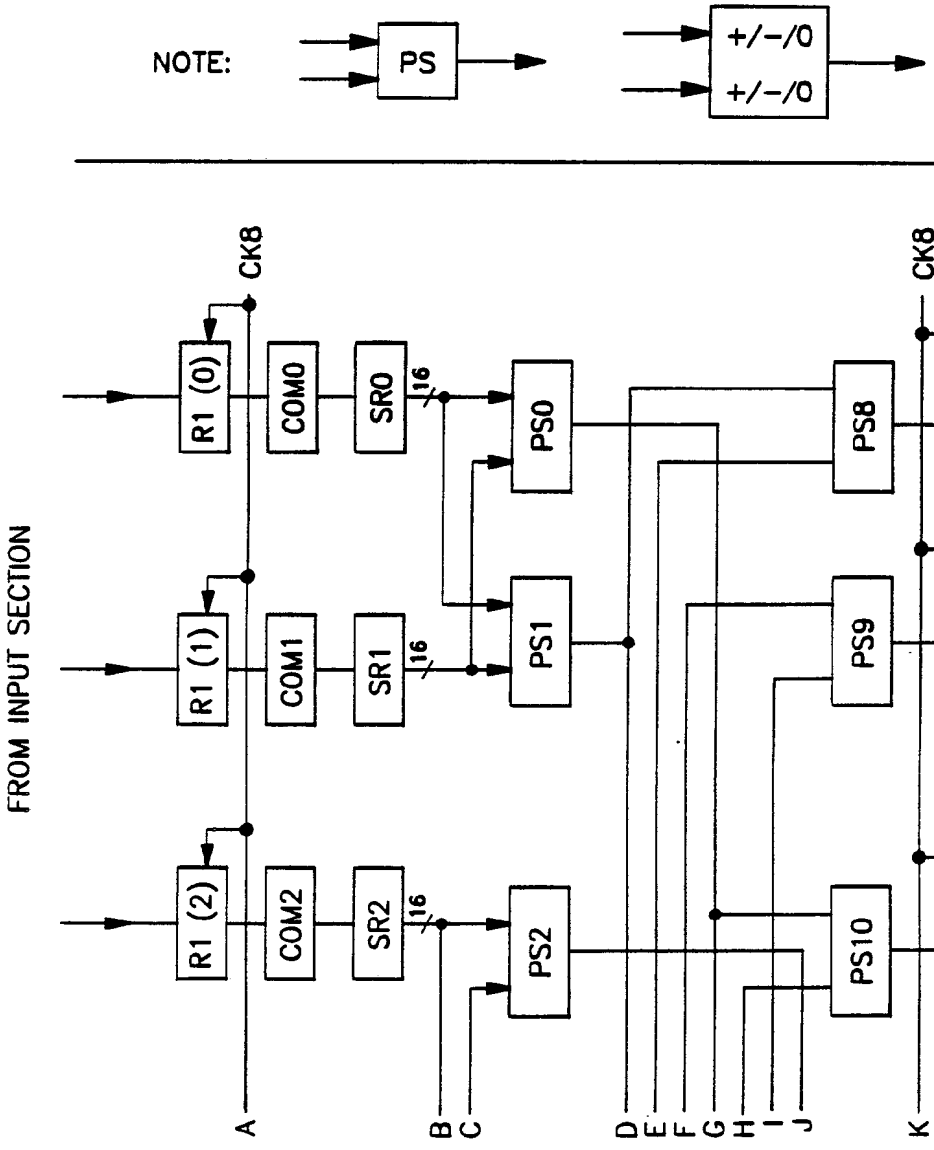


Fig. 2.13b

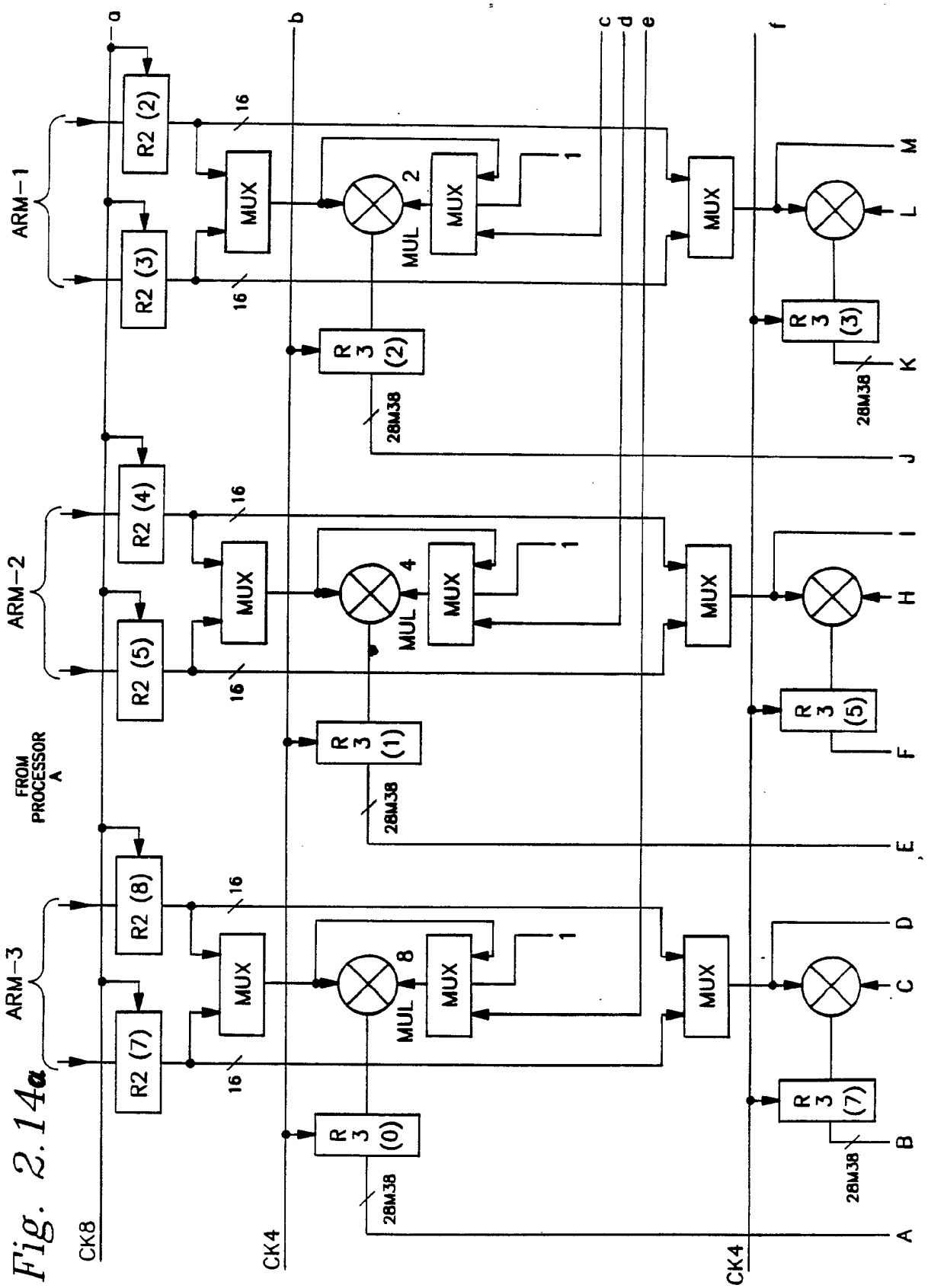

ARM-1
ARM-2
ARM-3
FROM PROCESSOR A
R2 (2)
R2 (3)
R2 (4)
R2 (5)
R2 (8)
R2 (7)
MUX
MUL
R 3 (0)
R 3 (1)
R 3 (2)
R 3 (3)
R 3 (5)
R 3 (7)
28M38
16
CK8
CK4
a
b
c
d
e
f
A
B
C
D
E
F
H
I
J
K
L
M

Fig. 2.14a

PROCESSOR – B

FROM AUXILIARY – INPUT SECTION

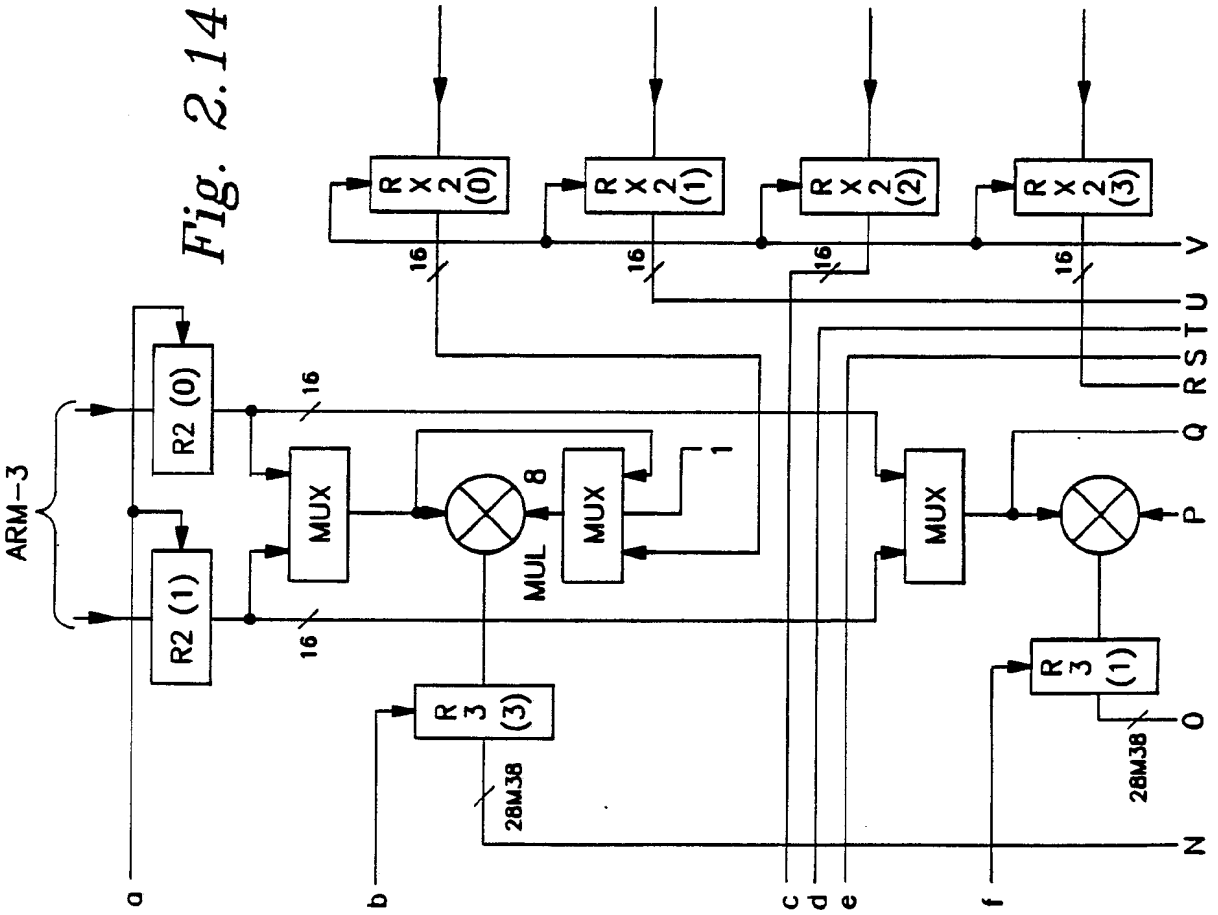


Fig. 2.14b

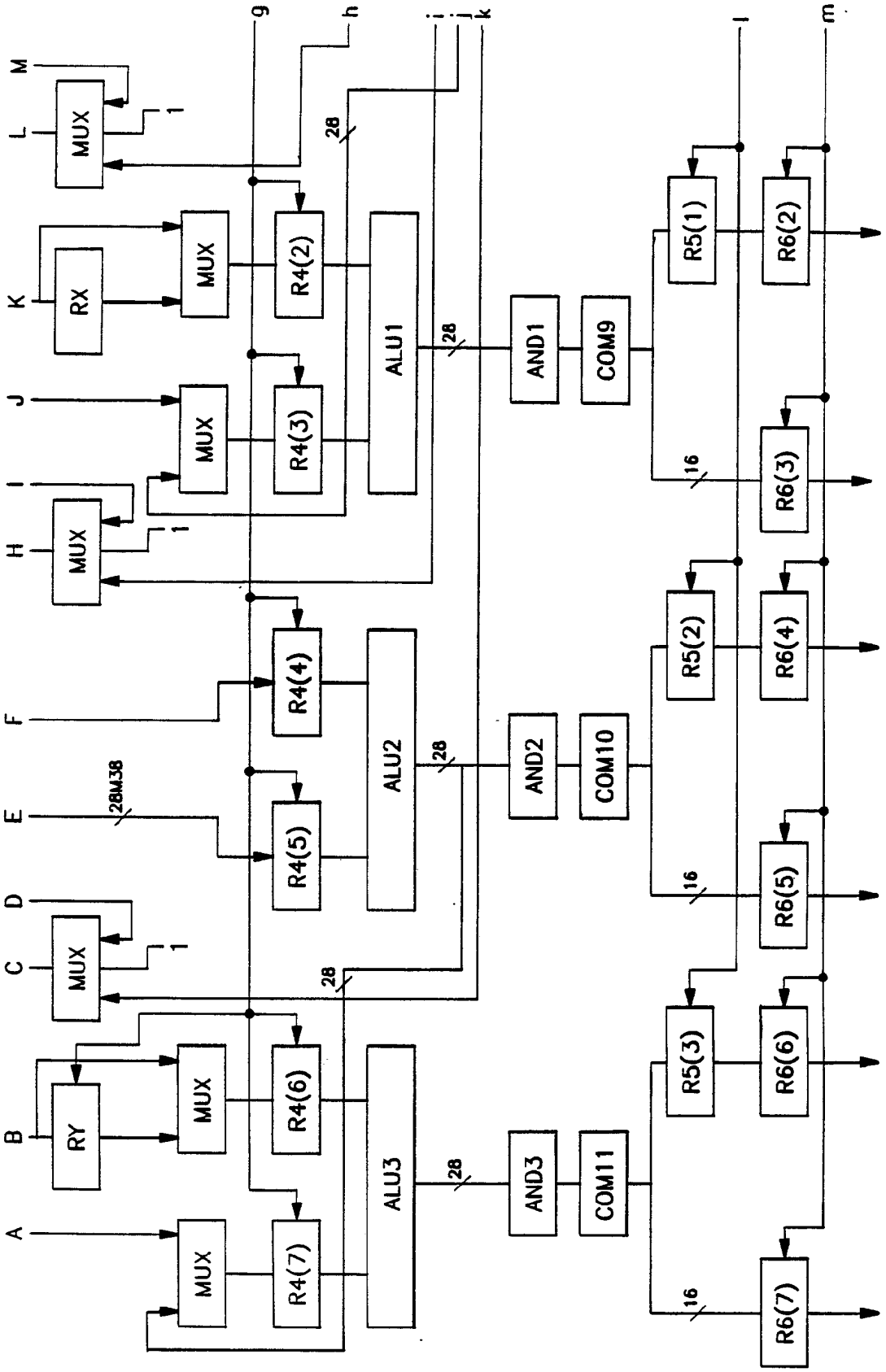
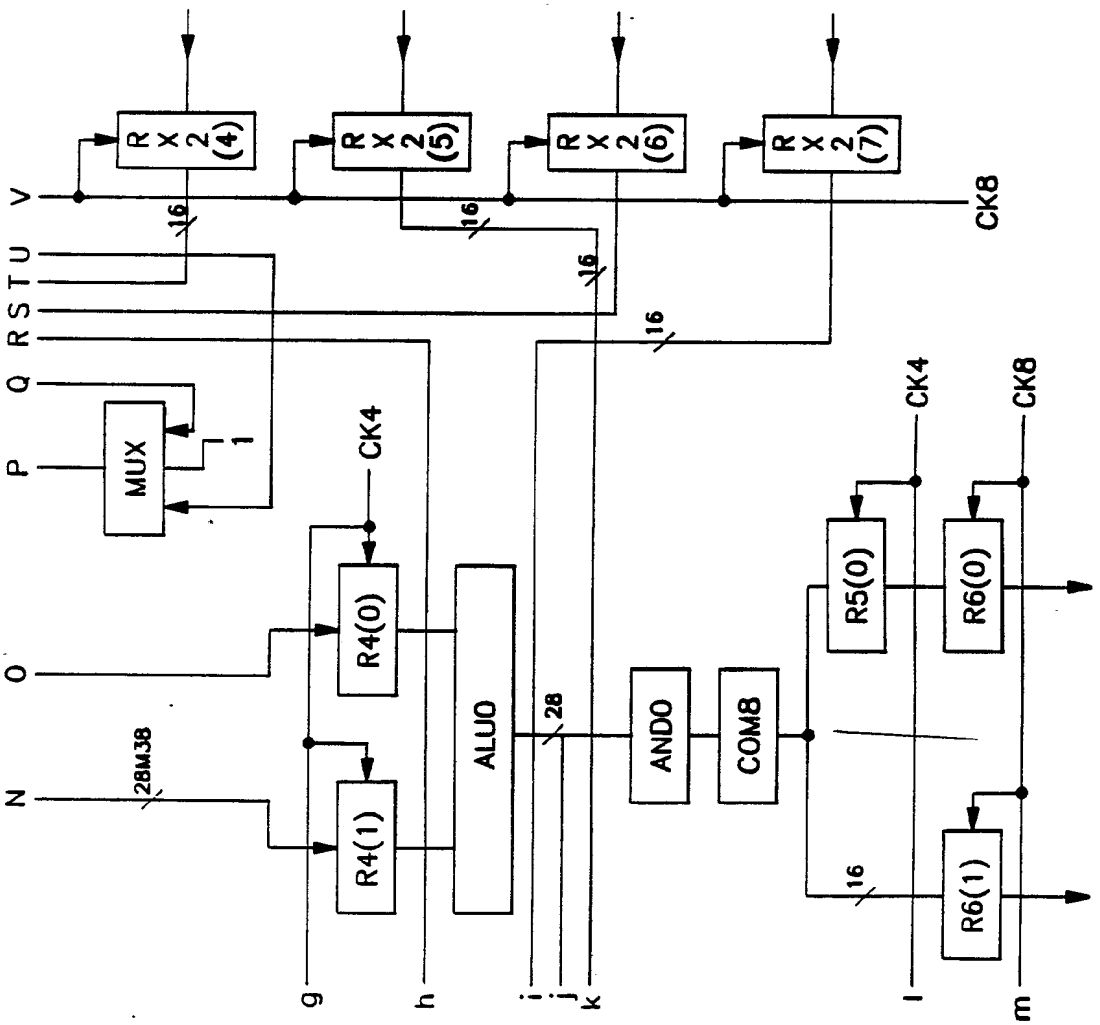

A
B
C
D
E
F
H
I
J
K
L
M
MUX
RX
RY
1
28M38
28
g
h
i
j
k
l
m
R4(2)
R4(3)
R4(4)
R4(5)
R4(6)
R4(7)
ALU1
ALU2
ALU3
AND1
AND2
AND3
COM9
COM10
COM11
16
R5(1)
R5(2)
R5(3)
R6(2)
R6(3)
R6(4)
R6(5)
R6(6)
R6(7)
TO OUTPUT SECTION

Fig. 2.14c

Fig. 2.14d

PROCESSOR – B

FROM AUXILIARY – INPUT SECTION



PROCESSOR B – TIMING

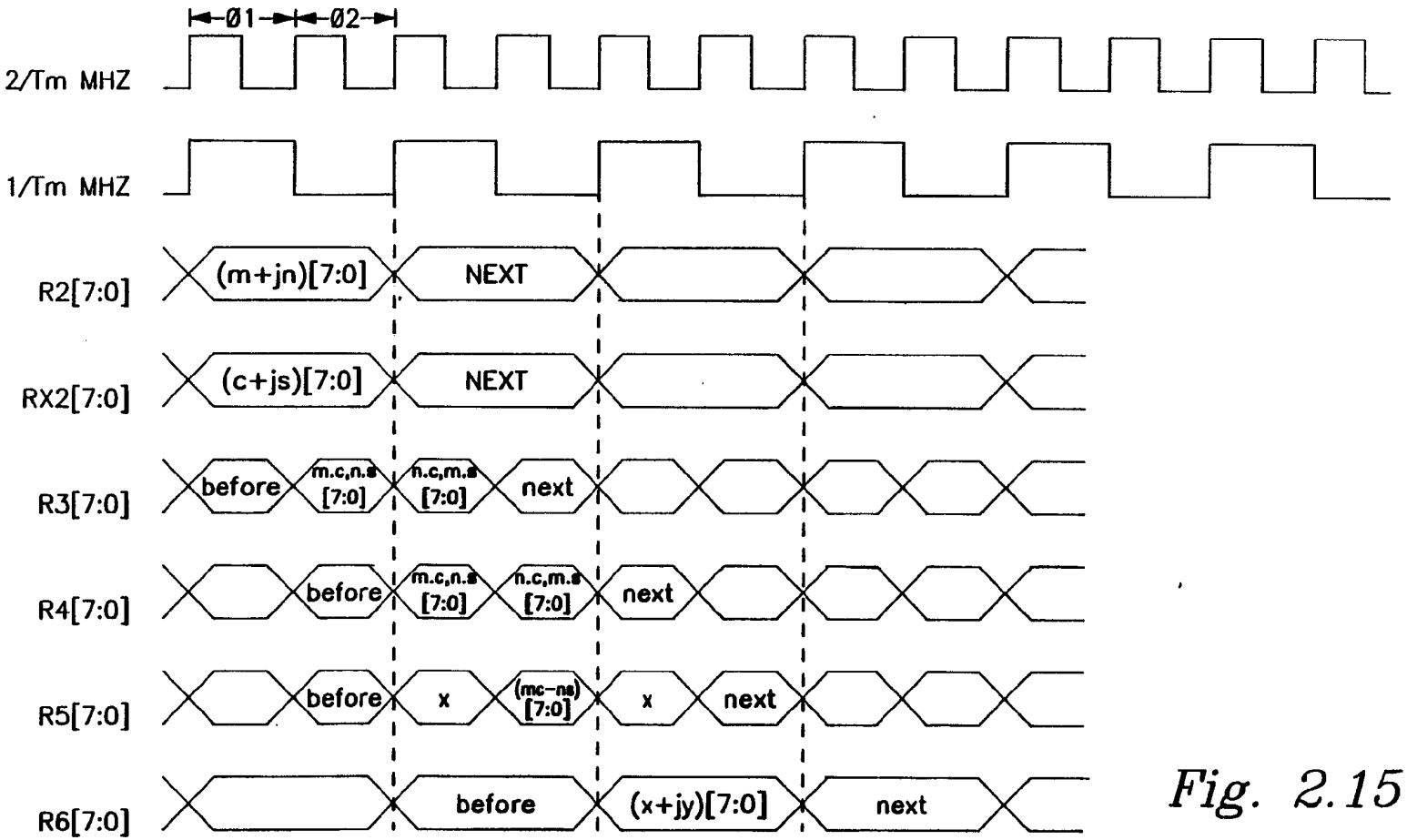


Fig. 2.15

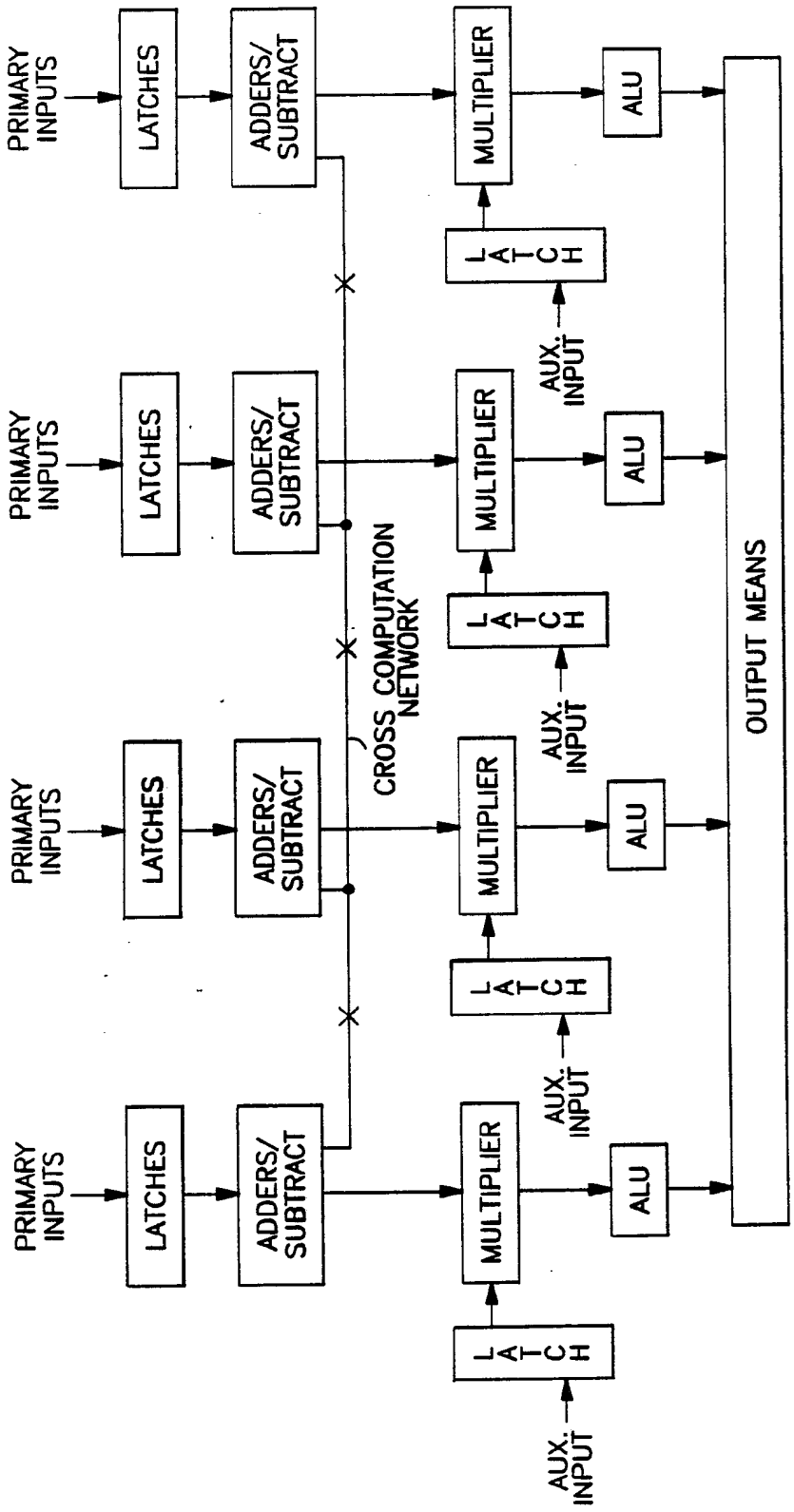


Fig. 3.1

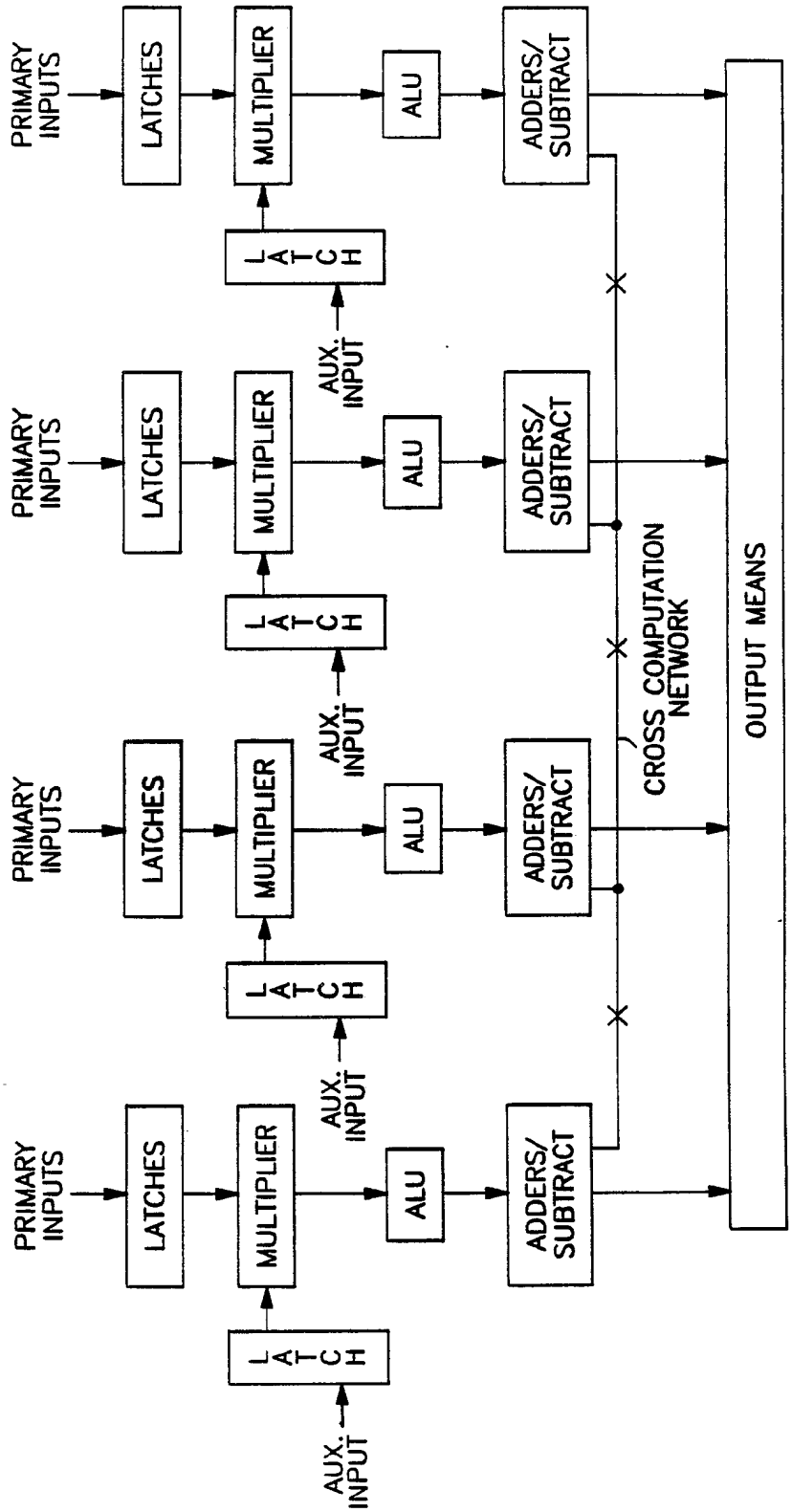
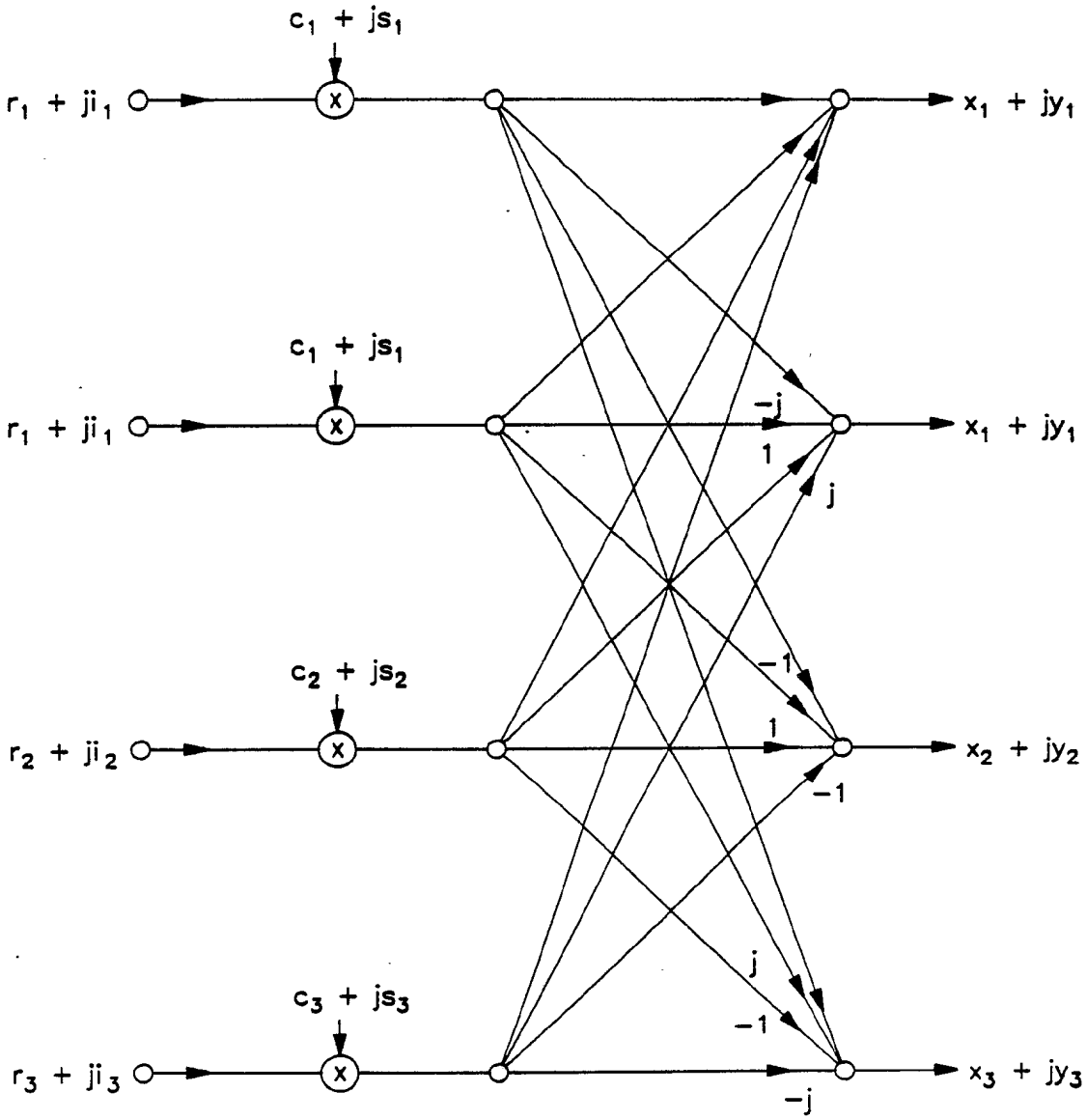

PRIMARY INPUTS
LATCHES
MULTIPLIER
ALU
ADDERS/ SUBTRACT
LATCH
AUX. INPUT
CROSS COMPUTATION NETWORK
OUTPUT MEANS

Fig. 3.2

Fig. 3.3


$c_1 + js_1$
$r_1 + ji_1$
$x_1 + jy_1$
$c_1 + js_1$
$r_1 + ji_1$
$x_1 + jy_1$
$c_2 + js_2$
$r_2 + ji_2$
$x_2 + jy_2$
$c_3 + js_3$
$r_3 + ji_3$
$x_3 + jy_3$
−j
1
j
−1
1
−1
j
−1
−j